Performance Modelling, and Adaptive Control for Linked Sequential Systems

Thesis

UNRESTRICTED.

Performance Modelling, and Adaptive Control for Linked Sequential Systems

A Thesis presented for the degree of Doctor of Philosophy

Paul Nicholas Morley

28 June 2012

Department of Design, Development, Environment, & Materials

Faculty of Mathematics, Computing & Technology

The Open University

UK

Date of Submission: 29 June 2012

Date of Award: 14 May 2013

Performance Modelling, and Adaptive Control for Linked Sequential Systems

Paul Nicholas Morley

A Thesis Submitted for the degree of Doctor of Philosophy

2012

Abstract

This thesis investigates the dynamics of linked sequential systems of machines in industrial laundries. Two aspects are considered: firstly the control of such systems and in particular the decision making point when a batch to be processed can be sent to one of many identical machines, and secondly the modelling of the whole system of linked machines.

The decision making point in the control of these systems is frequently implemented in a sub-optimal manner, or a manner which becomes sub-optimal as conditions change. An adaptive system is preferable and an Evolutionary Artificial Neural Network approach (EANN) is proposed. The EANN is tested on simulations of real laundry systems and shown to be effective. Then it is applied to two abstract game playing problems in order to better understand its limitations. Limitations are found to include the fact that if learning does not appear to take place, it is not possible to determine if this is a failure of the Evolutionary approach or the Artificial Neural Network parameters.

The dynamics and performance of Linked Sequential Systems in Industrial Laundries are not well understood or covered by theory in the literature. The theory of the performance of these systems is outlined, and an Agent Based

Model (ABM) simulation presented. The ABM simulation is explained and then the simulation is compared to a real world system in an existing laundry. The performance of the existing system is measured and compared to the prediction of the ABM simulation. The ABM simulation is shown to offer a better understanding of the system than the previous static calculation. Finally the ABM is used in a design exercise to show how it could be used to specify a system more accurately than the static calculation at design stage.

Declaration

The work in this thesis is based on research carried out at the Department of Design & Innovation, The Open University, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless explicitly stated in the text.

Acknowledgements

I would like to thank my supervisors, Professor Jeff Johnson, and Dr Anthony Lucas-Smith for their guidance and invaluable support through the period. I would also like to acknowledge support from my employer, Kannegiesser UK Ltd., in terms of access to information, particularly in respect of chapters 8-10. Finally and not least, I would like to thank Louise for putting up with me working through the night on many occasions, and my cats for their eternal support.

Table of Contents-

List of Figures

List of Tables

List of Acronyms Used

acronym	meaning	page first used
ABM	Agent Based Model	6
AI	Artificial Intelligence	5
ANN	Artificial Neural Networks	5
ART	Adaptive Resonance Theory	20
BP	Back-Propagation	16
C-ANN	Cascading Artificial Neural Network	21
EA	Evolutionary Algorithm	5
EANN	Evolutionary Artificial Neural Network	11
EBM	Equation Based Modelling	37
GA	Genetic Algorithm	26
GP	Genetic Programming	27
JSP	Job Shop Problem	3
MAS	Multi-Agent System	36
NFL	No Free Lunch (Theorem)	24
SME	Subject Matter Expert	7
SOM	Self-Organising Map	19

Chapter One: Introduction

I work in the field of industrial laundry as a designer for systems of linked machines. There is a very high capital cost of individual pieces of machinery (which are normally incorporated into linked sequential systems). There is a strong commercial drive to use machinery efficiently, and currently there is a lack of formal methods of modelling these systems, therefore the process of specifying these systems lacks rigour and on occasions systems will be over or under-specified with substantial cost penalties for the supplier and the operator.

Furthermore, laundry industry service providers normally operate with incoming work that is subject to a high degree of uncertainty, both in volumes and types of work. There are variations over both the short and the long term, and therefore a washing system must be flexible to deal with these variations. Generally however the variations are not observed and so a system that may have been specified well and optimised for a particular set of circumstances at its commissioning will not perform optimally when circumstances have changed. It follows that most systems in operation, are working sub-optimally.

From undergraduate work, including a supervised project for the Open University MEng degree (Morley, 2003), it was considered that there was an opportunity and a need to investigate the dynamics of linked systems of machines in industrial laundry systems so that such systems could be operated more efficiently and productively.

There are two threads to this research:

Firstly, the research investigates the control of linked systems of machines, specifically to investigate the application of techniques from Artificial Intelligence to the control of systems in order that they can be more adaptive to changing

circumstances. It was hoped that this would allow the development of control systems that could continually adapt to the short and long term variations of presented work so that the system could operate optimally (or near-optimally) for all its lifetime.

Secondly the methods of modelling linked systems of machines were investigated in order that a system could be specified with a higher degree of confidence in the early design stages. This means that the system would be more likely to be appropriate for its purpose and therefore be able to be operated optimally.

Figure 1 below shows a typical industrial laundry installation. This is of the loading side of two continuous tunnel washers (The 'Powertrans' model, manufactured by Kannegiesser GmbH).

Figure 1: Powertrans continuous tunnel washers

1.1 Context of the Research

The industrial laundry systems, like virtually all industrial processing operations, comprise a sequence of operations carried out in automated manner using machines linked in series (for example, as shown in Figure 2)

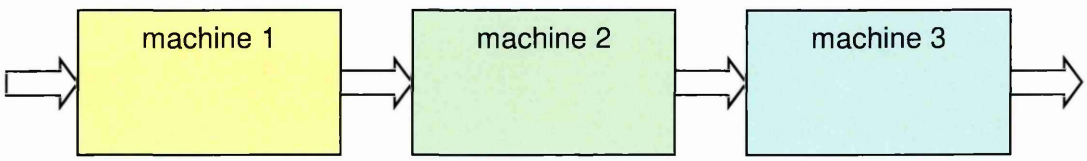


Figure 2: Linked sequence of machines

In practice, it is often the case that one or more of the functions carried out by one or more of the machines might be carried out by multiple machines in parallel. This is shown in Figure 3 below.

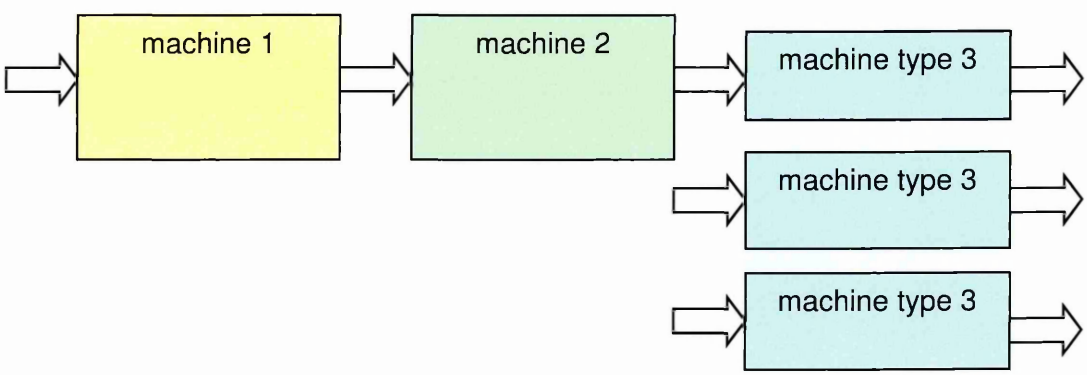


Figure 3: Linked sequence with one function by parallel machines

Clearly, in this situation a choice has to be made when laundry work comes out of machine two, and has to go to one of the machines of type three.

This thesis investigates the dynamics of linked systems of machines in the specific field of industrial laundry systems.

Figure 4 shows a typical washing system as used in this field. Here, the workflow is from right to left. The sequence of operations is from washer to press to dryer, with the two washer-press combinations operating in parallel. When work comes out of one of the presses, a travelling conveyor receives the work and transfers it to one of the seven dryers. This is the point at which a decision has to be made as to which dryer. This is a variation of the Job-Shop Problem (JSP). (Garey, 1976)

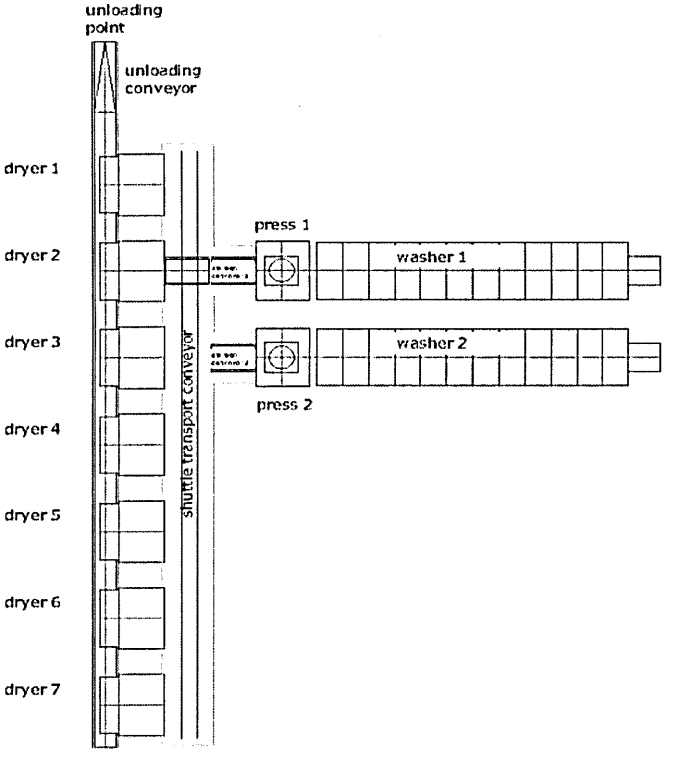


Figure 4: Typical layout of washing system

The problem of controlling and modelling such a system is not trivial. Firstly the number of different states of the whole system is so high that it is impractical to

systematically define the absolute best next course of action, and secondly, the mix of work being processed by the system tends to change in both the short and long term. It was found that existing controls are generally static and do not adapt to changing circumstances. Therefore performance tends to decrease over time. This is described in more detail in chapter three.

Within this field there is a continual commercial drive to ensure the optimal use of such systems. Additionally as concerns grow about the unsustainable use of resources and energy, there is further impetus to ensure that systems are utilised efficiently, with minimal wastage of resources or energy.

Furthermore, as equipment becomes ever more complex and therefore expensive to invest in, the need increases for equipment suppliers and operators to avoid over or under investment in equipment, which then proves insufficient or under-utilised. Therefore there is a growing need for a better prediction of the performance of potential systems at the design stage. Currently the methods of modelling such systems are very simple mathematical equations. In practice, when designing systems a rough calculation is done, and then the system is over-sized according to some established heuristic to allow for uncertainty. Consequently it is never known if the system has been efficiently specified.

1.2 Research Aims and Research Questions

The aims of the research were:

1 To establish if paradigms from Artificial Intelligence can be used in the control of linked sequential systems in industrial laundries, in particular in a decision making capacity, in order to improve the performance of such systems and their tolerance to variation.

2. To establish new ways of modelling linked sequential systems in industrial laundries, in order to improve the accuracy of prediction and control, and thereby improve their design.

Following undergraduate work and a preliminary literature review into the paradigms of Artificial Intelligence (AI), these aims were developed into the following research questions:

Research Question 1.

Is it viable to apply the AI paradigms of Evolutionary Algorithms (EAs) and Artificial Neural Networks (ANNs) to the control of linked sequential systems in industrial laundries, in particular at a decision making point?

If so, can specific strategies be identified for their implementation, in order to develop better methods for such control and thereby improve the performance of such systems?

Research Question 2.

Does Agent Based Modelling offer an effective approach to the simulation of linked sequential systems in industrial laundries?

If so, is this a better method of simulating such systems than the existing methods, and can it offer a better method of predicting the performance of such systems in order to improve the specification of such systems at the design stage?

1.3 Research Methods

The research methods chosen had to be appropriate to the resources available. Resources available included computing resource and some programming knowledge, as well as access to a wide range of industrial laundries, plus existing knowledge of their design specifications. Key limitations included time and resources for any larger scale experimentation.

The research methods applied to these questions were firstly to carry out experimentation into the application of Evolutionary Algorithms and Artificial Neural Networks to the decision making point of linked sequential systems in industrial laundries and analogous situations, in order to determine the validity and effectiveness of this approach.

Secondly, an Agent Based Model (ABM) simulation of a real world linked sequential system was designed. Its results were compared with the actual

performance of the system in order to establish its credibility in predicting performance of such systems.

1.4 Why were Artificial Neural Networks (ANNs) and Evolutionary Algorithms (EAs) investigated?

During a preliminary literature survey into established paradigms in AI, it was found that there was widespread agreement as to the general advantages and disadvantages of these methods. (See for example (Hecht-Nielsen, 1989), (Hopgood, 2000), and (Picton, 1994),)

ANNs are well suited to making classification decisions from noisy data. They are good at interpolating in a data set (although less good at extrapolating from that set).

ANNs are easily described by a linear sequence of numbers and hence are easy to combine with an EA.

EAs are non-deterministic and are useful in circumstances where there is a very large search space, especially where the search space is not yet defined. They are well suited to self-adapting systems and in situations where circumstances change and where continual optimisation is required. An example is given in (Nolfi & Parisi, 1997).

In choosing to investigate these paradigms, rule-based systems were considered and discounted as they require a Subject Matter Expert (SME) to provide the rules. This is in fact the case with current industrial laundry control systems. A key drawback is that once the rules are set, it is not practical to continually revisit

them as circumstances vary – quite often because it is not known that the circumstances have varied. Therefore the performance of the controlled system tends to decrease due to the effects of changing circumstances.

Fuzzy logic was also discounted as it also requires the initial establishment of a set of probability variables with truth values. Again, once these are set it is not practical to revisit them as circumstances vary. Hybrids of Fuzzy logic with other paradigms have been established in the literature – for example (Koprinkova-Hristova, 2010) which describes the use of an ANN to adjust the parameters of the fuzzy rule set. While this approach has been shown to be successful in the control of highly non-linear plant, the application does require a plant mathematical model in order to provide the ANN with an error to be minimised. Such a mathematical model was not available for the linked sequential systems under consideration in this research.

1.5 Why was Agent Based Modelling (ABM) used?

During a preliminary literature survey, it was found that ABM was well suited for simulating linked sequential systems because such systems comprise many interacting but autonomous parts. Each of these parts usually has simple rule-based deterministic behaviour, but the performance of the system overall is an emergent property – i.e. cannot be predicted by examination of the behaviour of the component parts alone.

Such a model compares favourably to reality in that it doesn't require simplification of data, or reduction of heterogeneous data to averages.

An ABM can be validated at microscopic level – by examining the behaviour of each agent, but also at macroscopic level – by comparing the behaviour of the overall model to that of the real world system.

Finally, ABMs can be relatively simply scaled up by introducing more agents, and the behaviour of each agent can be changed without the need to change overall system level equations.

It was considered therefore that ABM offered a good match to the requirements of a simulation of linked sequential systems in industrial laundries. In particular this was considered to be a better match than the alternative system dynamic paradigm (Forrester, 1961) which models a dynamic system through a set of differential equations. These require the homogenisation of data, cannot be validated at microscopic level, and because the differential equations operate at system level, cannot be easily modified as the modelled system changes.

1.6 New Knowledge

This research has led to a new way of analysing the dynamics of linked sequential systems in industrial laundries. This development of the Agent Based Dynamic Model simulation is novel in the field of industrial laundry, and is a marked improvement on the current static approach. The model is a new and effective method of understanding and predicting the performance of such systems, and offers significant advantage over the current methods.

The mathematical analysis of the calculation of performance of both tunnel washers and dryer systems presented here (sections 3.3 and 3.4 respectively) is developed in a more rigorous manner than any previous literature.

The research into application of AI methods to adaptive controls has added to existing scholarship in these methods and contributed to the overall depth of knowledge: specifically by application to different situations not previously tried. Also, the method of combination and application of EAs and ANNs is novel.

During the course of this research the following research papers were published:

- "Training a Genetic Algorithm and Neural Network Hybrid Pattern Classified by Population Statistics". AISB, 2005. (this reports on some of the work described fully in chapter 3)

- "Application of EANN Hybrid to run a Conveyor Control System". AISB, 2010. (This reports on some of the work described fully in chapter 4)

- "Evolving Neural Networks To Play Noughts and Crosses". ISKE, 2009. (This reports on some of the work described fully in chapter 6)

These papers are included as appendices A-C respectively

1.7 Thesis Structure and Summary

The following is an outline of this thesis:

1.7.1 chapter 1 - Introduction

1.7.2 chapter 2 – A review of the state of the art

The thesis begins with a review of the state of the art in the field of hybrids of ANNs and EAs. These are reviewed individually first and then as hybrids - Evolutionary Artificial Neural Networks (EANNs). ABMs are also reviewed and their application to simulation of complex systems.

1.7.3 chapter 3 – Industrial laundry

The field of industrial laundry is described, with washing systems and existing static methods of specification and modelling.

1.7.4 chapter 4 - An EANN unsupervised classification system

An early series of experiments was carried out in this area, and was reported on in (Morley, 2005). It is shown empirically and qualitatively that the use of a GA can be effective in generating an ANN to approximate a classification function, which distinguishes between classes, where a training set of pre-classified samples were not available, and the only information given a priori is a known proportional split between samples.

1.7.5 chapter 5 - EANNs for industrial control

This chapter describes an experiment into the application of an EANN (similar in principle to that from chapter 4) to an industrial control problem. A conveyor system is modeled where a destination decision has to be made. Part of this work has been previously published. (Morley, 2010)

1.7.6 chapter 6 - EANNs for problem solving I

Following the work described in chapter 4, the EANN approach was taken further and applied to a game playing scenario. This was because the industrial control problem was considered to be similar to many games in that a pattern of inputs (current state) requires a control system to make a decision and provide a single output (next state). The application to a game was thought to offer a way of understanding these control systems better. In the case of the work reported on in chapter 5, this did not lead to a successful system.

1.7.7 chapter 7 - EANNs for problem solving II

Following chapter 6, the game was simplified and in this case the control methodology was more successful. Part of this work was previously published (Morley, 2009).

1.7.8 chapter 8 - an Agent Based Dynamic Model

This describes the work done to develop an Agent Based Dynamic Model to simulate linked sequential systems in industrial laundry.

The dynamic model has added to the understanding of these systems, and can be used to commercial advantage. It provides a more accurate specification of

systems and a higher degree of confidence in the system so specified, and it allows a prediction of the effect on smaller changes in the system. For example - if a single machine had a fault that led to a reduced level of performance, the model would give an understanding of the effect of that single fault - which may be of help in deciding whether to commit resources to remedy that fault.

1.7.9 chapter 9 - verification and validation of the Agent Based Dynamic Model

This describes field based research of an operating laundry with a highly automated linked sequential system. Actual operational parameters were obtained, along with actual quantitative performance data for the system. The parameters were set into the Agent Based Dynamic Model and a simulation run. The results obtained showed the effectiveness of the model.

1.7.10 chapter 10 – design exercise

Following verification of the ABM model, it was used in a design exercise, to demonstrate how it could be applied in a live design process, and to demonstrate the type of outputs.

1.7.11 chapter 11 – conclusions & recommendations

To complete the thesis this chapter reviews the original research aims and methods and considers conclusions that can be drawn from the entirety of the work and answers to the original research questions.

Chapter Two: A Review of the State of the Art

In this review, two key paradigms of AI - Neural Networks, and Evolutionary Computation – are reviewed. These are then brought together in the field of EANNs. Then the field of Multi-Agent Systems is described.

2.1 Artificial Neural Networks (ANNs)

ANNs are a mature AI paradigm and were considered promising to examine in order to research adaptive controls. This was because in very general terms ANNs have the ability to cope with noisy or imperfect data

For a decision point in a linked sequential system, it is usual to have a very large number of data inputs to the control system, but only one output (a decision on what to do). Effectively this becomes a pattern recognition problem. The relationship between the inputs and the outputs is in principle deterministic, but often the relationship between the inputs can be complex and impossible to be recognised.

Over decades of research, many different types of network have emerged, although common to them all is that the network is composed of multiple identical processing elements which take a number of weighted inputs, and carry out some mathematical function to give an output. This approach is also referred to as Parallel Distributed Processing.

Table 1 contrasts some key differences between sequential and ANN computing.

sequential (Von Neumann) computers...	**ANN computing...**
possess a central processor that every instruction has to go through	no central processor - all processing is distributed around the network
instructions executed extremely quickly	each unit can be quite slow, but the speed of the system comes from the massive parallelism
a failure is usually catastrophic	a failure of a single, or few units leads to a drop in performance rather than complete failure (the so called 'graceful degradation' property)
have to be programmed explicitly	can be trained by example
a programme run can be checked step by step	no easy way of checking the results (a 'black box' answer is given)

Table 1: Key differences between traditional & ANN computing

2.1.1 Early History of ANNs

The first development of ANNs was by inspired by biological neural systems. (McCulloch & Pitts, 1943) described a biological neuron as being a circuit element that 'fires' when the sum of its multiple inputs exceed a threshold. This element could therefore be modelled by a simple mathematical equation, and was shown to be able to perform Boolean logic.

ANNs are useless without being able to learn - adapt in order to perform a useful task - and a key milestone was (Hebb, 1949) which stated;

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells, such that A's efficiency, as one of the cells firing B, is increased."

There is a form of positive feedback, so that when cells A & B fire together, then the connection between them is strengthened and in future A becomes more effective at firing B. This has become known as 'Hebbian Learning', and was the first learning process that was described mathematically. It is also known as 'Coincidence Learning'.

Early systems (for example ADALINE (Widrow & Hoff, 1960)) were single layer systems. A key realisation was provided by (Minsky & Papert, 1969) who demonstrated that single layer processing units were unable to implement non-linearly separable problems, for example the XOR problem. However (Rumelhart & McClelland, 1986) showed the ability of multi-layer ANNs to implement non-linearly separable functions, and also demonstrated the generalised delta rule (known as Back Propagation - BP) which could be used to train a multi-layer ANN using examples. This is known as Supervised Training, where a set of patterns of inputs, and the associated required output is known. Then the training set of patterns of inputs are presented to the network, and the weights adjusted so that the actual output corresponds to the required output.

BP is also known as the Delta Rule, and is a gradient descent error minimization process. Effectively the change for each weight is proportional to the error between the actual output and the desired output for a set of training patterns.

2.1.2 General Forms of ANNs

ANNs are described as either feed-forward or recurrent (Picton, 1994). A feed-forward network, has a simple layer structure, with each layer receiving inputs from only the previous layer, and giving outputs only to the subsequent layer. Figure 5 below shows this general scheme.

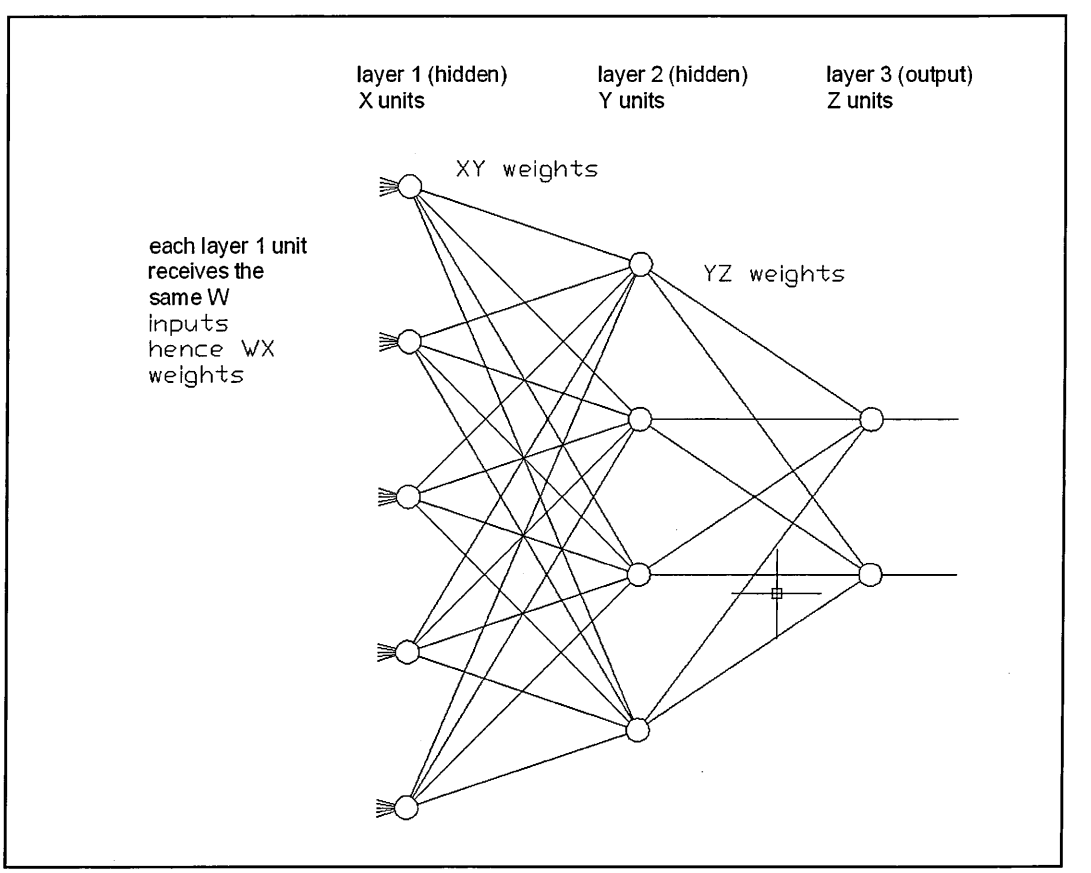


Figure 5: General form of feed-forward ANN

A recurrent network, is one where the output of a unit can be routed back to give an input to a previous layer (or its own layer). Hopfield networks are a major type of recurrent network. These networks typically have complex dynamics, and once a set of inputs is applied they take a time before they stabilise (if in fact they do stabilise at all).

Traditionally the weights associated with the links between each neuron are found by a back-propagation algorithm (BP), which is essentially an error minimization (hill climbing) algorithm. This is effective but requires a Training Set of pre-classified data samples which can be used to train the network (set the weights) and then a separate testing set (Picton, 1994). A software tool in Java script to implement this is described by (Panoiu, et al., 2011) and shows the widespread use of this paradigm.

It has been shown (Cohen, 2003) that adding extra unclassified data samples after supervised training tends to reinforce accurate training or reinforce inaccurate training.

A feed-forward ANN, such as shown in Figure 5 above, is able to approximate any function. This result was first shown by (Kolmogorov, 1957). The Kolmogorov Existence Theorem states that a two layer network is able to implement any measurable function to an arbitrary degree of accuracy. This was shown in a manner more practically applied to ANNs by (Hornik, et al., 1989). This means that for *any* mapping function, there exists a network that can approximate this function to the level of accuracy required.

2.1.3 Other General Forms of ANNs

There are other forms of ANNs that are important to mention because they form a significant part of the literature, but they are not described in detail because they were not used in this research. These are:

Self-Organising Maps

Self-Organising Maps (SOMs) (Kohonen, 2001) are a type of ANN, trained using unsupervised learning. They take as their input high-dimensional data and the individual elements arrange themselves in a topology that is derived from the input data. This in practice gives a low-dimensional representation of the input data. They are often used in situations where they assist with representation and visualisation of complex data. They do not tend to be used in control applications. SOMs were not considered further in this research because they do not give a simple output and are therefore not suited to a decision-making control process

Hopfield Networks

Hopfield (Hopfield & Tank, 1982) developed the Hopfield Network, which is a feedback network. As with Kohonen SOMs they are not designed to give a simple decision making output and are therefore not suited to control applications. They are a form of associative memory. During training, given a pattern of inputs, the network (typically using Hebbian learning) can adapt the network element weights so that it can serve as a content addressable memory. That is – if in non-training mode part of the same pattern is presented – then the network can recreate the whole pattern.

(Wang, 2006) gives a current application of Hopfield Networks for finding an optimum solution to a geometric problem. (Liu, et al., 2006) presents theory behind the periodicity of the solution to generalized Hopfield networks. Hopfield Networks were not considered further in this research because as with SOMs they do not give a simple output and are therefore not suited to a decision-making control process

Adaptive Resonance Theory networks

Grossberg's Adaptive Resonance Theory (ART) gives a network that fuses bottom-up data driven classification with top-down expectation driven classification. A classification network is split into a comparison part (which contains a set of memories of different possible classes) and a sensory part which takes sensory data and compares against the possible (expected) classes. A parameter (known as the 'vigilance parameter') determines how close a match will be required in order to classify the detected data as belonging to a particular class, or if it should be stored as a new class.

ART networks were not considered further in this research because the complexity in their implementation did not lend itself to experimentation with available resources.

Cascading Artificial Neural Networks (C-ANNs)

C-ANNs are a form of ANN which begins with a minimal network and during the training phase the network automatically adds new hidden units one by one creating a multi-layer structure. They do not use BP and typically learn very fast, (Fahlman & Libiere, 1991).

Their main downside is that they are often too sensitive to training data – they over-fit the data (they model a function which is more complex than the original function being modelled, with the extra complexity leading to a better fitting to the noisy training data, but a worse fit to the original underlying function), leading to poor generalisation to new unseen data. For this reason they were not considered further in this research.

Evolutionary ANNs (EANNs)

The coupling of Evolutionary Algorithms with ANNs was considered to be an ideal method for this research, and is described in the next section.

2.1.4 Summary of Different Forms of ANNs

Table 2 below compares and summarises the advantages and disadvantages of the different forms of ANN considered.

	Advantages	Disadvantages
Feed-forward ANN	Simple to implement Generalise well within the extents of their training set	BP can be slow to converge Training data required Do not generalise well beyond the boundaries of the training set
Kohonen SOM	Unsupervised training	Not designed to provide a single decision making output
Hopfield Networks	Unsupervised training	Not designed to provide a single decision making output
ART networks	Effective for classification including self-adaptation for new classes	Complex to implement Designed for classification rather than control decision-making
Fast BP	Faster to converge than standard BP	Complex computationally May be unstable in convergence May be task-dependent i.e. not widely applicable without substantial re-development
C-ANNs	Self-adapting	Susceptible to over-fitting
EANNs	Can be self-optimising Once optimised, the ANN operates like a normal ANN – i.e. simple computation, fast results, simple decision making process	Can take a long time to evolve. Effective evolutionary parameters can be difficult to find.

Table 2: Comparison of forms of ANN

2.1.5 Developments of BP

In the 1990s, the Backpropagation algorithm (BP) was developed further. The main downside of standard BP is that it can be slow to converge on a solution. Some different variants of this (Cho, et al., 1991) include:

- Standard BP with gain
- Improved BP
- BP with adaptive gain

Speeding up BP is usually achieved by increasing the learning rate. This can be interpreted as increasing the gradient of descent in the hill-climbing analogy. The effect is to make convergence less certain and some forms of fast BP are susceptible to overshoot, or convergence instability. They frequently fail to converge in a finite time.

(Cho, et al., 1991) proposed a method of automatically re-initialising the element weights (with random values) when convergence speed becomes too slow. In the hill-climbing analogy this is like restarting in a different part of the landscape.

An example of an application of enhanced versions of BP is given in (Kim, 2002). Different versions of BP were used – based on adding momentum factors to reduce training time and susceptibility to finding only local minima - and also quickpropagation (Fahlman, 1988) which assumes the error surface is (locally) quadratic and by computing the second order derivative of the error surface, jumps directly to the minimum of the parabola. Here ANNs were used in image recognition for a traffic control problem.

The downside of these modified methods is that they are usually very complex and must be adapted to suit a particular application. This is an example of the implications of the No Free Lunch theory (Wolpert & Macready, 1997)

Various methods of backpropagation for dynamic neural networks are described in (deJesus & Hagan, 2007), and ANNs are widely used across a number of fields. For example (Kuo, et al., 2011) describes a method of using ANNs with BP to estimate evaporation rates in paddy fields using meteorological factors. However, the fundamental nature of BP remains, which is that it relies upon training data to provide an error which can then be minimised. Therefore it was decided not to use BP in this research.

2.1.6 No-Free-Lunch (NFL) Theorem

All algorithms that search for an optimisation of a cost function perform exactly the same, when averaged over all possible cost functions, (Wolpert & Macready, 1997).

This means that, over the set of all possible problems, each search algorithm will do on average as well as any other. For an algorithm that performed particularly well on a certain problem or problem class, then NFL Theorem states that its performance on other problems will be worse, so that on average its performance will be the same as a hypothetical algorithm that performed with exactly the same effectiveness on every possible problem. This is illustrated in Figure 6 below.

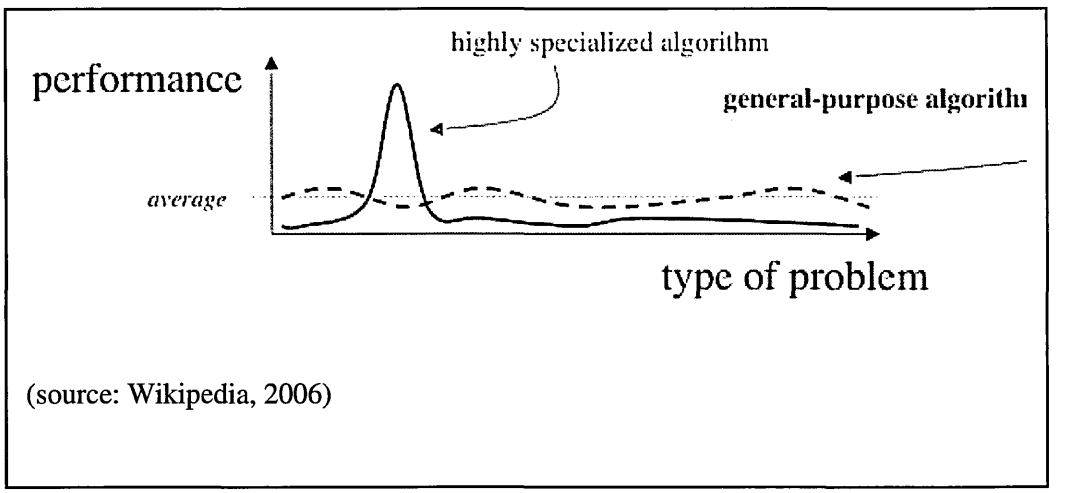


Figure 6: No-free-lunch theorem

2.1.7 Conclusions

From this review of the field of ANNs the following conclusions were drawn:

- The Kolmogorov Existence Theorem and also the results of (Hornik, et al., 1989) state that for any requirement, an ANN does exist. Therefore it is not futile searching for it with an EA.
- BP relies upon training data in order to adjust weights and train an ANN. This is not ideal in circumstances where training data may not be available, and this includes changing circumstances where initial training data may become out of date. Therefore BP was not used in this research and another method of finding the necessary weights for the ANN was required.
- EAs are a useful way of finding the weights of an ANN and have been practically applied.

- The No-free-lunch theorem regarding search algorithms (for example EA), states that just because an algorithm may be effective at one problem, it does not necessarily mean that it will be effective for another.

2.2 Evolutionary Algorithms (EAs)

Evolutionary Algorithms (EA) or Evolutionary Computation, are umbrella terms for any computation strategy that draws inspiration from biological (Darwinian) evolution and these strategies include Genetic Algorithms (GA), Evolutionary Programming, and Evolution Strategies. The literature does not always use these terms consistently, and it can be difficult to define these precisely and unambiguously. However, Evolutionary Computation can be recognised by the use of a population based approach, some method of evaluating individuals within the population against their suitability for some reason (a fitness function), some method of producing more individuals based upon the more fit members of the existing population, and iterative progress towards a goal.

EAs provide a powerful way of exploring a complex solution space. Essentially an EA depends on being able to describe a system by a sequence of symbols - by analogy: a chromosome. Different systems' chromosomes can be split and combined to create a new generation. Some form of fitness function is then used to select the 'best' individual systems and these go forward to create the next generation and so on. A random operation is also usually introduced, analogous to genetic mutation.

There are many variations to this approach; the operators used can vary; for example, mutation & recombination are two common operators inspired by

biological genetics, but are not the only ones possible. Equally, there are many different approaches for managing the population; for example the strongest instances in one generation may be retained in the next, or the entire population may be replaced with new instances.

An EA will not guarantee to find the optimum solution, or even any solution. However, they are effective in homing in on some effective solution, and are especially useful in extremely large search spaces.

2.2.1 Terminology

There is overlap and inconsistency of use of terminology, but the following is a summary of the major techniques:

Genetic Algorithms (GA): possible solutions to a problem are encoded numerically as sequences of numbers (chromosomes) and these are recombined and mutated to produce a population of possible solutions which are then evaluated against a fitness function

Genetic Programming (GP): As GA but computer programs are represented as a tree structure, and genetic operations are applied to the trees rather than a numerical sequence.

Evolutionary Programming: As GP but the program is fixed and it is the parameters of the program which are allowed to evolve rather than the structure of the program.

Neuroevolution: This is an application of EAs to train ANNs. The term is used for both the evolution of the link weights in an ANN, and also for the evolution of the structure of the ANN (or both).

2.2.2 Genotype - Phenotype, and Ontogeny

The following concepts are borrowed from biological genetics.

- **Genotype:** the genetic encoding of an organism - the chromosome. Mutation, cross-over, and other operators act on the genotype modifying it.
- **Phenotype:** the actual realisation of that genotype as the appearance and properties of the organism
- **Ontogeny:** the process of development of the organism, i.e. the link between genotype to phenotype and the interactions with the environments

By biological analogy: genotype is the cellular DNA, but the phenotype is the organism that results from the development of that DNA. These concepts are summarized in Figure 7 below (after (Banzhaf, et al., 1998). The figure has been extended further by adding in the influence that the phenotype has over heredity through the competition for resources and selection for reproduction – 'survival of the fittest' - plus the influence that phenotype (and also, the overall group of phenotypes - the population) has over the next generation of phenotypes; 'culture'. Cultural learning has been used by (Curran & O'Riordan, 2004) to evolve ANNs for problem solving.

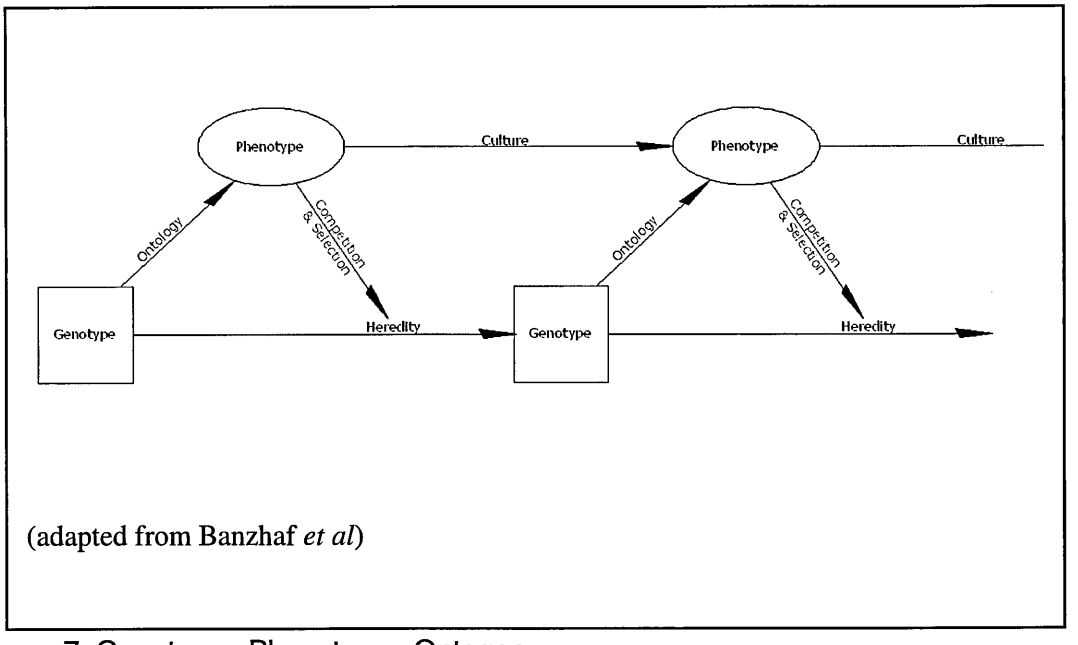


Figure 7: Genotype - Phenotype - Ontogeny

2.2.3 The Early History

In 1954 Barricelli first used computer based evolution in order to play a simple game (Barricelli, 1962). It was not until the 1960s that the field became more widely known. (Bremermann, 1962) reported work that included features recognisable as GA.

Rechenberg and Schwefel in the 1960s developed Evolutionary Strategies for problem solving (Schwefel, 1977), using a population of problem-specific solution representations which were selected based on fitness and then mutated.

The field became more widely known following (Holland, 1975). This laid down the basic framework for evolutionary systems and outlined the key features. It also outlined Holland's Schema theorem.

A schema is a string sequence in a chromosome of some fixed length. For practical purposes it makes most sense to use the term schema to describe some subset of a chromosome which has specific properties in its own right. (Clearly the validity of this statement depends upon the method of encoding the chromosome – it will be very relevant to a tree representation, where a schema could be a whole branch of the tree representation, but less relevant in other methods of encoding).

The schema theorem gives a probability that an individual schema will survive to a specified generation (and if it will proliferate). Clearly a schema that is short, has a greater chance of surviving to the next generation as it is less likely to be affected by either mutation of crossover. Equally, a schema that significantly contributes to the fitness of the overall solution encoded by the chromosome will also be more likely to survive.

Schema theory has influenced later work – for example (Poli & McPhee, 2003), (Poli & Langdon, 1998), and (Poli, et al., 2004). However, while schema theory is an important part of the history of EAs, and is important in gaining an understanding of how the evolutionary process works, it has not been considered further in this research because it was not considered critical in achieving the research aims. It is a viable question for further work to determine how schema theory applies to the EANNs investigated in this research.

2.2.4 Evolutionary Programming

Fogel developed the Evolutionary Programming technique (Fogel, 1966). This has been applied to many real world problems. For example, (Fogel, 1993) describes Evolutionary Programming to create ANNs that can play noughts &

crosses. (Hopgood, et al., 2004) describes an evolutionary search to find optimum parameters in a very large search space for an industrial process. However as EP is a programming technique, it is not well suited to finding the weights in an ANN and so is not used in this research.

2.2.5 Variation of EAs from the Biological Analogy

The advantage of computational evolution is that it does not have to match biological evolution. For example: while biological evolution uses mutation and crossover, EA is free to use mutation only, single or multiple point crossover, or other mathematical operators.

With biological evolution the parent generation must die. EA is free to retain the fittest members of the parent generation as well as the offspring generation, and the strategy employed for replacing the population members can vary according to the problem. For example, it might be appropriate to replace the entire population every generation, or it might be appropriate to replace only a small percentage.

Biological Evolution is Darwinian. Characteristics are passed from one generation to the next through the genetic code. EA can explore the use of alternative mechanisms, for example Lamarckian Inheritance (where characteristics developed in the lifetime of an individual can be passed to the next generation). This only makes sense in the context of an EA where individuals have some method of optimisation other than merely EA – e.g. a hybrid EA with hill-climbing optimisation. Similarly cultural learning can take place. For example (Curran & O'Riordan, 2004) which describes a method of applying an EA to a decision making role in a game playing situation, although this is in many ways analogous

to the control application of linked sequential systems which is the focus of this research.

2.2.6 EA Applications and Hybrids

Various hybrid systems using EAs and GAs have been described in the literature. EANNs (see section 2.3) are well established. Other schemes include EAs with other soft computing paradigms; for example (Sharma, et al., 2012) describes an application of GA with Fuzzy Logic to analyse the reliability of multi-unit systems, where the GA was used to calculate parameters of the optimisation model of the system prior to the use of Fuzzy Logic to compute performance measures of the system. This example shows the flexibility of the GA approach and its adaptability to different applications.

A similar example is (Yogeswaran, et al., 2009) which describes the use of a GA with simulated annealing algorithm (SA) to solve a machine loading problem. Here SA is used to improve the operation of the GA by modifying the best and worst chromosome in the population. The application of the GA to a machine scheduling problem (which shares characteristics with the problem of controlling linked sequential systems in industrial laundries) shows that this method in principle is established in the literature (although not for this specific application).

Another EA application to a scheduling problem is given by (Wisittipanich & Kachitvichyanukul, 2012). Various EA approaches were applied to the Job Shop problem (JSP) which is known to be Non-Polynomial-complete (NP-hard) and shares similarity with the control problem under investigation here.

An EA requires the choice of many parameters such as: population size, method of crossover and mutation factors. (Manikas & Godfrey, 2011) describes the exploration of different parameters for GAs applied to the JSP and found that the best results were given by using single-point crossover, and that the results were not sensitive to mutation rates.

2.2.7 Conclusions

From the above review of EAs, the following conclusions were drawn:

- EAs are simple to implement (although can be computationally intensive and slow to run). They are useful for complex problems, and do not need a sophisticated understanding of the problem in order to be applied. They are practical to implement using resources available to this research. In particular they can be tolerant to the setting of their parameters, although that may make the search process take longer
- EAs are difficult to rationalise – their performance cannot necessarily be explained. They do not scale up well (curse of dimensionality), and it is often not clear where to stop the evolutionary process. They cannot hill-climb: once an individual has been generated that produces a good result, the EA does not allow for small adjustments to that individual in a simple hill-climbing manner to optimise its performance. However, this is catered for in hybrid methods which combine EA to generate a population of solutions which are then modified within limits, to find an immediately better solution. This approach was not however used in this research, due to the constraint of computing and programming resources.

Following the above comparison of EAs, it was decided to research further the use of EAs to evolve ANNs, and apply this to the control of linked sequential systems.

2.3 Evolutionary Artificial Neural Networks (EANNs)

EAs have been applied to both the problems of finding link weights of an ANN, and finding the an effective structure of a network, with schemes reported which alternatively evolve only the structure of the network (leaving the problem of finding the weights to a deterministic approach - such as classical back-propagation: a gradient descent optimisation algorithm) or keep the structure fixed and evolve the weights. (Yao, 1999)

For example, (Palmes, et al., 2005) describes an algorithm used to evolve both the weights and the structure of ANNs. (Angeline, et al., 1994) describes a system of evolving both weights and structure. (Fogel, 1993) is an earlier example which also used evolutionary programming to evolve both the weights and structure. This latter case is particularly interesting as it refers specifically to a system where the selection process acted on only the output of the algorithm and not on the ideas underlying the output. A point made in (Fogel, 1993) is that this is effective and efficient, contrary to the view put forward in (Penrose, 1989)

EANNs then, are a special class of ANN where EAs are used on a population of candidate ANNs to evolve one of the following;

- the connection weights (on a fixed network structure)

- the architecture of the network (but not weights - typically the weights would be determined using a classical algorithm such as BP)
- simultaneous evolution of both architecture and weights
- other parameters, such as the learning rules, the transfer function, etc.

When applied to ANNs, there is a key difference between BP and EA. BP is a gradient descent algorithm, which means that it cannot be used if the error function is non-differentiable. Conversely, EAs can be used successfully if the fitness function or error function is non-differentiable.

2.3.1 Encoding Scheme

However the EA is implemented, it is necessary to decide how to encode the ANN in a chromosome. There are two main encoding methods. Direct encoding where the chromosome will explicitly include everything about the network including the link weights, and Indirect encoding where the chromosome will only include information about how to build the network, and then link weights must be found separately (typically if the EA is used to develop the ANN architecture only, the ANN will then be trained using, say, BP to learn the weights)

In this research it was decided to use direct encoding, as this leads to a simpler method of implementation (BP does not have to be implemented) and a simpler method of encoding – the chromosome is simply a string of numbers representing the link weights (in order).

2.3.2 Critical Evaluation of EANNs

It was considered that EANNs were well suited to the control of linked sequential systems in order that they can adapt to changing circumstances and therefore operate optimally over their full operating lifetime. This was because:

- ANNs are well suited to dealing with noisy or imperfect input data.
- ANNs are well suited to taking a large number of inputs and giving an output that is effectively a 'black box' decision at a key decision making point in a process.
- for reasons outlined in section 2.1.7, BP is not appropriate for finding the link weights in this context, and for reasons outlined in section 2.2.7 EAs were considered appropriate for this requirement.
- EAs are simple to implement and require modest computational resources

EANNs were considered a promising method for controls of linked sequential systems as the EA aspect can allow the optimisation of the system in the short term (to generate an ANN that can be used to actually control the system), and the EA can then be run in parallel to the system over the longer term with a comparison between the EA generated ANNs and the ANN actually used for control. When an ANN has been generated that outperformed the ANN actually used for control then the controlling ANN could be supplanted. This process could go on permanently, hence the system would continuously optimise itself and adapt to changing circumstances.

2.4 Agent Based Models

An Agent Based Model (ABM), or Multi-Agent System (MAS), is a method of modelling complex behavior using individual agents which operate according to their own set of rules, rather than having the whole system operate under some overarching control.

A Complex System is one characterized by high dimensionality, non-linearity, and the emergence of macroscopic properties from microscopic behavior and in particular the interaction between the microscopic behavior of different elements of the system. As such Complex Systems are very hard to model using traditional scientific methods which rely upon the isolation and variation of individual parameters.

As the microscopic behavior of system elements can often be described with confidence, the rules governing the behavior of the modelling agents can be fairly easily specified.

In Complex Systems the emergent behavior can be fairly stable (as opposed to Chaos Systems where the emergent behavior is highly sensitive to initial conditions). This adds to the usefulness of ABMs in modelling Complex Systems because once the behavior of the agents is specified, the initial conditions are relatively unimportant.

Agent Based Modelling can be described as a bottom-up methodology, as opposed to the traditional top-down methodology. The comparison between these is clearly illustrated by Figure 8 below.

The traditional static approach is generally defined as an Equation Based Model (EBM) approach. The behavior of the whole system is defined by a set of (typically) Ordinary Differential Equations. (Parunak, et al., 1998) describe ABM as giving rise to the emergent properties of the whole system from the interactions between individuals, while EBM models the system observables through the use of high-level equations.

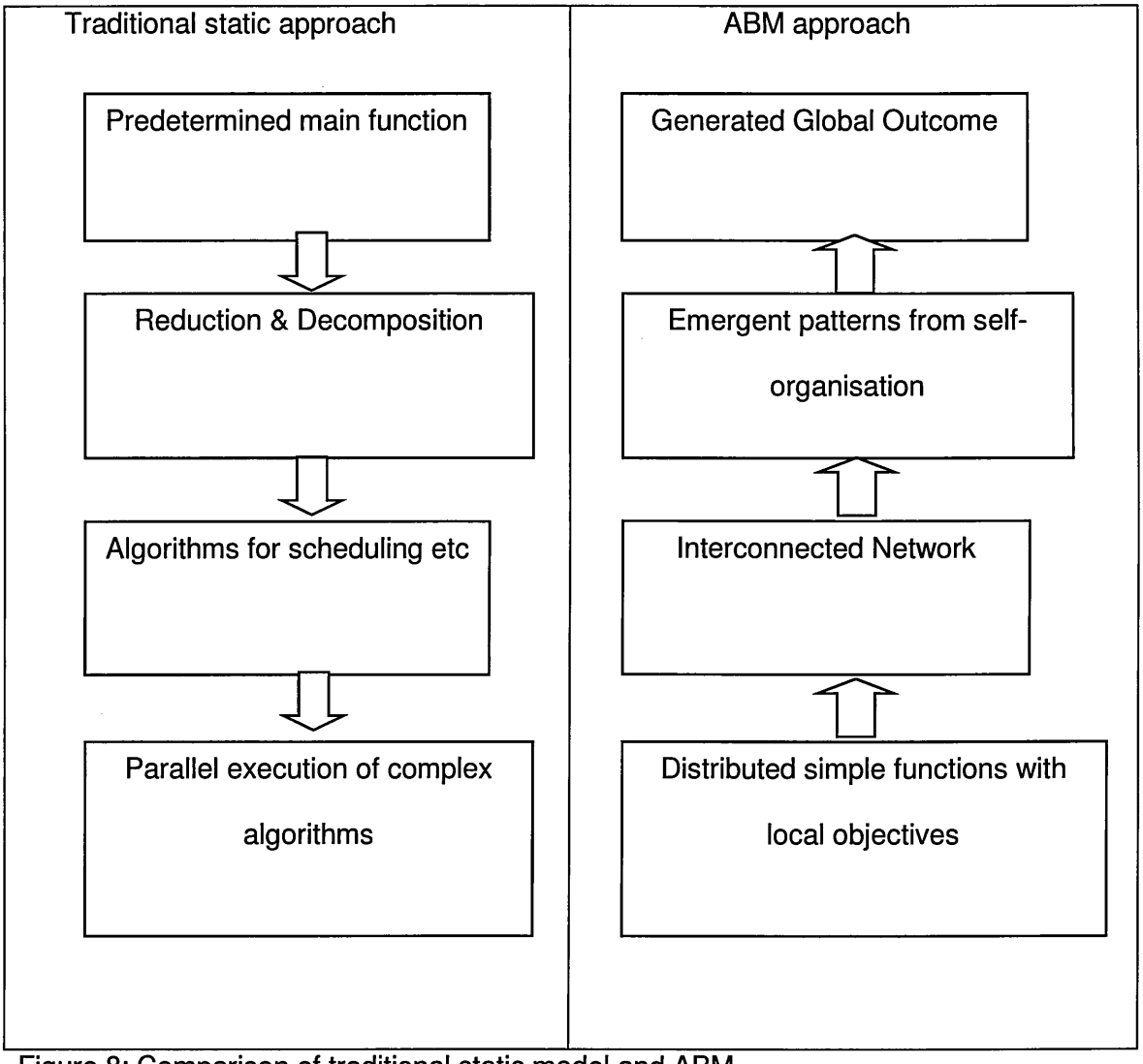


Figure 8: Comparison of traditional static model and ABM

from (Reaidy, et al., 2003), quoted in (Nilsson & Darley, 2006)

2.4.1 Agents

(Wooldridge, 1997) gives the following definition for an agent:

An agent is an encapsulated computer system that is situated in some environment, and that is capable of flexible, autonomous action in that environment in order to meet its design objectives.

In relation to the elements of a machine system (such as that of Figure 2) each element is situated in the whole system which forms its environment, and has well defined links of information and actual work between different elements. As the system is a built system of machines in the real world, each element is clearly persistent, and under this control paradigm each element control has the autonomy to make its own decision about how to operate. Therefore this system fulfills the criteria as being constructed from agents.

(Hopgood, 2000) gives the additional requirement that each agent have some form of intelligent behavior. In industrial laundry systems, some elements may require or benefit from some intelligent control, but others (for example - simple conveyors) may only have very simple behavior.

2.4.2 Relevant Applications

ABMs and MAS have been applied in a number of fields. For example Distributed Artificial Intelligence control of electricity distribution - ARCHON (Jennings, 1996) (ARchitecture for Cooperative Heterogeneous ON-line systems)

(Jennings, et al., 1998) go on to describe further applications in Air Traffic control, Manufacturing and Process Control, Telecommunications, Information Filtering & Information Gathering, ecommerce, Games and Interactive Entertainment

Application to a Packaging Company

(Nilsson & Darley, 2006) describes the application of an ABM in a packaging company characterized by rapidly changing customer demands, no genuine understanding of the relationships between customer order patterns, factory capacity, machine speeds, and other parameters, and a requirement for a 'virtual factory' to test impacts of policy changes on customer service levels and costs. This set of circumstances is very analogous to the industrial laundry field which is the focus of this research.

In this case the packaging company had differentiated itself from the competition by offering a more flexible service at high quality levels, but there was no clear understanding of the impact of different operational strategies upon costs and logistics,. Moreover the operation was too complex to allow simple modelling, with parallel decision making in different parts of the company, different performance measurements made in different parts of the company, and a commercial need to optimize use of resources.

An ABM was developed in the following sequential manner:

i. Process mapping and information gathering from the local managers and staff – from this an initial model was developed
ii. Developing a more detailed model and calibrating it based on real data

iii. Verification and validation

iv. Actual modelling and simulation of different scenarios

The agents used were 9 machines (linked with both sequential and parallel organization), sales, operations planning, warehouse, and customers. Each agent was fairly simple in design, represented by logical rules.

Nilsson & Darley observed that an advantage of ABM is that the model validation can be done microscopically, with each agent's behavior being validated, and then macroscopically, with the emergent properties of the whole being compared against historic operations.

Following development, the company was able to use the model for simulations to aid decision making scenarios such as the company's largest customer significantly increasing orders but production capacity being already near maximum (therefore how should the company accommodate the increased demand).

Nilsson & Darley conclude that ABM is applicable for systems which are dynamic, distributed in time and space, made up of many interacting and autonomous parts (agents), where there are several objectives and conflicting constraints, and where emergent phenomena could be exhibited. These factors hold for industrial laundries.

This was a very relevant study and naturally led to the question: Could this approach be used for the simulation of production systems generally, *before* their design, as an aid to specifying the system required, and then predicting its performance?

This question was not addressed in (Nilsson & Darley, 2006) but is addressed by this research (chapters 8-10).

Application to a manufacturing cell environment

(Renna, 2011) describes the use of a MAS scheduling system in a dynamic environment of manufacturing cells. The development of a simulation in order to implement and evaluate the approaches is also described. The MAS approach was successful and the simulation environment could be used as an aid to management decision making, but again the simulation was not used for performance prediction as a design tool. A similar study was reported in (Ruiz, et al., 2011) for the use of a MAS simulation for intelligent manufacturing and warehouse management. (Nejad, et al., 2011) also reports on a MAS for the control of a manufacturing environment, focusing on the protocols for negotiations between agents.

Application to a Production Line

(Barbosa & Leitao, 2011) describe the application of ABM to a production line. The production line also involved linked sequential machines. In this case, the production line comprises 10 separate steps, with one (number 7) being carried out by one of two parallel work stations. The following key conclusions are derived from this:

- the simulation can be used to verify that the system is in normal operation – well balanced
- the simulation can look at the effect of one of the parallel work stations being out of operation

This reinforced the conclusion above – can this approach be used for simulation of the production system before its design as an aid to specifying the system required?

2.4.3 Critical Evaluation of ABMs

The previous sections have found that ABMs are ideally suited for the modelling of linked sequential systems. Each part of the system can be modeled as an agent, and the interaction of the agents can effectively simulate the system. As previously stated, once the behavior of the agents is specified, the initial conditions are relatively unimportant.

ABMs are relatively simple to implement. The behavior of a single machine (therefore a single agent) is usually easy to specify. The complexity of the system comes from the interactions between the agents. However in a linked sequential system the interactions between machines are usually straightforward to model. Therefore an ABM can usually be constructed more easily than a working out all the equations governing the behavior of the overall system for an EBM.

ABMs are also easy to amend compared to the EBM method. If a machine in a linked sequential system is changed for a machine with a different behavior (or if a machines performance changes or it is out of operation) then the equations in the EBM have to be altered as a result. With an ABM it is easy to modify the behavior of a single agent leaving others unchanged. Equally, it is easy to take away or add in more agents to reflect modifications to the system.

For these reasons ABM was considered the best method with which to model linked sequential systems.

2.5 Conclusions

Following to this review of literature, it was decided that the problem of *controlling* linked sequential systems could be usefully addressed using EANNs, because in a linked sequential system there is a large amount of data gathered from the system that represents the current state, from which a decision has to be made on the next action, but the data is often to be noisy and difficult to interpret. There are also commercial and operational reasons why human intervention is not ideal for optimisation, and so self-optimisation is best, although there is no need for explanation of the optimisation process. EA can run parallel to the operation process and the level of computation involved is not problematic in comparison with the normal speeds of production in industrial laundry.

It was also decided that the problem of *simulating* a linked sequential system in an industrial laundry could be addressed using an ABM, because the components of a linked sequential system fulfil the criteria to be viewed as agents, and because their individual behaviour is usually simple and deterministic with the system behaviour as a whole being an emergent property. The advantage therefore of the ABM approach is that the behaviour of each individual element can be tested and verified, and should that individual element be changed (or its control be made more complex) then it is relatively easy to scale the ABM to account for this.

Chapter Three: Industrial Laundry

3.1 Overview

As explained in section 1.1, industrial laundry was considered to be a useful field for research because it commonly uses linked sequential systems of machines, where utilisation of the whole production line is important, and the combinatorial complexity of the systems makes it impossible to predict the performance of the system without running a simulation. Therefore there is a benefit to an improved method of simulating such systems. It is also subject to short and long term unpredictable fluctuations of work to be processed – therefore it is suited to developing control systems that are able to adapt to changing circumstances.

Industrial scale laundries today process between 300,000 and 3 million pieces per week. They typically specialise in hospital work or commercial work (for hotels and similar). They may also process either flatwork (bed-linen, table-linen, towels, blankets) or garments (theatre work and staff uniforms for hospitals, or rental staff workwear for commercial operations - for example, overalls for factories). (Ferron, 2011)

The laundry industry is a low cost enterprise and is a high user of unskilled labour. Nonetheless, labour costs are a high percentage of the overall costs, and thus there is a requirement for automation but with capital costs as low as possible.

It is the washing and drying machines – principally continuous tunnel washers and the ancillary systems of dryers and conveyors - that are of interest to this research.

3.2 Sources of Variation

The work to be processed by laundry systems will vary. The variation of the work will only rarely be anticipated, and generally will not be known at the time or even retrospectively. There are three classes of reason for variation.

Properties of the linen being processed

The pieces of linen themselves will vary – due to textile manufacturing tolerances, and due to differing ages of linen in a population having different levels of wear and tear, and therefore weights and sizes.

The linen presented for laundry will range from dry to soaking wet which also affects the weight. They may also be presented in range from virtually clean to extremely soiled, affecting weight, colour, and handling characteristics.

There may be foreign objects mixed within the linen.

Variables such as this may account for short term variations (random effects, or the effect of for example a large batch of new linen being introduced into the system).Short or medium term variations may be introduced by, for example a period of bad weather leading to a higher percentage of linen being returned to the laundry wet.

Long term variations may be caused by a gradual shift in linen specification. For example a laundry business may change to a higher (heavier) specification of towel, but replace stock only organically, thus the change would be very gradual over a period of years.

Properties of the customer base served

Consider a laundry which specialises in hospital work, and serves many (as many as 100) separate hospitals from a wide geographic area. The mix of work being presented may contain up to 200 different types of work - for example, under the sub-heading 'sheet' there will be single sheet, fitted sheet, counterpane, cot sheet, theatre sheet, stretcher sheet, and all these require slightly different types of processing.

Linen coming from different hospitals may be different. Some hospitals specialise in certain types of surgery, or infections, and therefore their linen mix will be different to a general hospital. Equally, different hospitals may have different policies relating to bed changing. If a laundry company loses or gains a contract to supply one hospital then its overall mix of work may change as a result. Over time, many such small changes may occur, the overall change being significant.

Laundry Strategy

It could be that a laundry company takes a conscious decision to increase a particular type of work. This might be due to the relative profitability of different types of work, or the availability of the processing assets. Their sales force can be targeted appropriately. This can have a short term effect on work mix (when a new contract starts, the effect can be a step change), or a long term (the cumulative effect of many contract changes).

When the laundry is specified and designed, the machine types and quantities will be calculated, installed and commissioned according to the projected work

types and mix, with an allowance for uncertainty. However, the factors given above are examples of factors that may affect this.

Normally the effect of work mix variation is to reduce processing efficiency. It follows that most laundries are processing at less than optimal efficiency for most of the time.

3.3 Continuous Tunnel Washers

The workhorse of a modern laundry is the continuous tunnel washer (Rogers, 2003). This machine is a long tube, internally an Archimedean screw, which takes work in batches approximately every 2 minutes. The work is loaded at one end, and it travels through the tunnel being subjected to different wash processes. It is unloaded at the back of the tunnel clean, whereon it is pressed to extract the water and then sent by conveyor to one of usually several dryers.

Figure 9 below, shows a typical machine. This machine is built into a barrier wall which delineates the dirty/clean areas of the laundry. This view is of the loading end, with the loading hopper underneath the loading bags which each carry 50kg of laundry. (This is a similar machine to those shown previously in Figure 1)

Under fully-automated control the machine will call for a bag of work typically every 2 minutes (the actual cycle time is variable), so producing 1500kg per hour. Larger and smaller batch size machines are available, and a laundry may have several machines.

The batches are kept separate inside the machine, and different wash programmes (different water flows, temperatures, and chemical treatments) used for each, according to classification. This allows the most energy efficient and environmentally friendly wash process to be used. Therefore there is an advantage in having the laundry separated into types, before washing. Within the laundry industry, it is a widely used term, to define the act of classifying, or separating into types, as 'sorting'.

After the washer there is a press to extract loose water, and then a shuttle conveyor to transport the pressed-batch to one of a number of dryers (see Figure 4). The interaction between the linked sequence of machines is a variation of the Job-Shop Problem, which is known to be NP-complete (Garey, 1976) and for which EA approaches are known to be effective at finding solutions, (Khuri & Miryala, 1999).

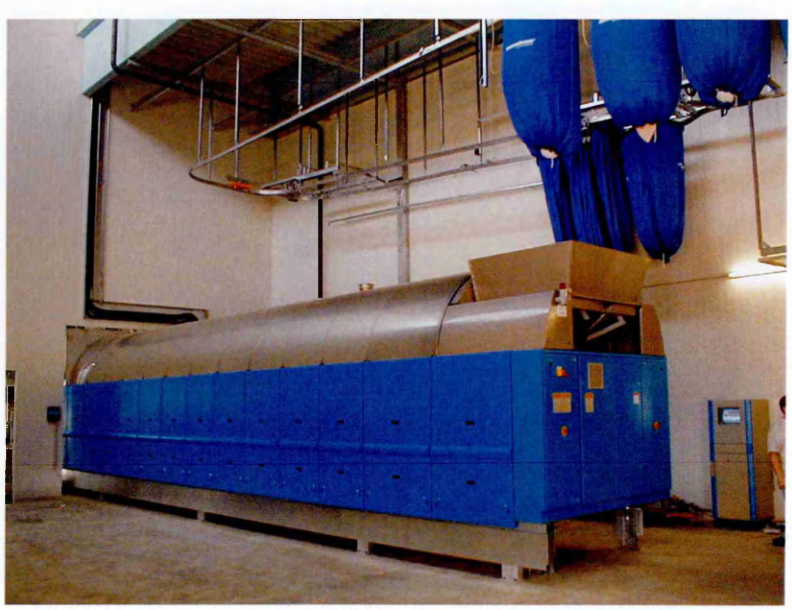


Figure 9: Continuous tunnel washer

The production rate of a continuous tunnel washer is calculated as follows:

(Kannegiesser GmbH, n.d.)

L load size (individual batch size - standard from 25kg to 100kg)

i_o overload factor (typically 10% of L)

N number of compartments

C cycle time (seconds)

U utilisation factor (actual production as a % of maximum)

$$P_{max} := \frac{3600}{C} \cdot L \cdot (1 + i_o)$$ equation 1

Equation 1 gives theoretical maximum production, assuming maximum loads produced per hour (according to the cycle time - no idling time) and that every load is the full load size including tolerable overload.

$$P_{av} := \frac{3600}{C} \cdot L \cdot U$$ equation 2

Equation 2 gives the actual expected production, assuming the average load size has no overload, and allowing for a typical idling time - 15% of uptime is common.

Washing time[1] (time spent in the washing machine)

$$T_{wash} := C \cdot N \qquad \text{equation 3}$$

For example, consider the operation of a 14 stage 50kg continuous tunnel washer, with a 2 minute (120 second) cycle time:

theoretical maximum production (from equation 1);

P_{max} = (3600 / 120) x 50 x 1.1

= 1650 kg/hour

actual expected production (from equation 2);

P_{av} = (3600 / 120) x 50 x 0.85

= 1275 kg/hour

washing time (from equation 3);

T_{wash} = 120 x 14

= 1680 seconds

= 28 minutes (this is a typical wash-time)

[1] note - time spent inside the washing machine is a key factor in wash quality and in a continuous tunnel machine, a time between 22-35 minutes is typical depending on the type of work and degree of soiling. 27 minutes is customary for hospital work.

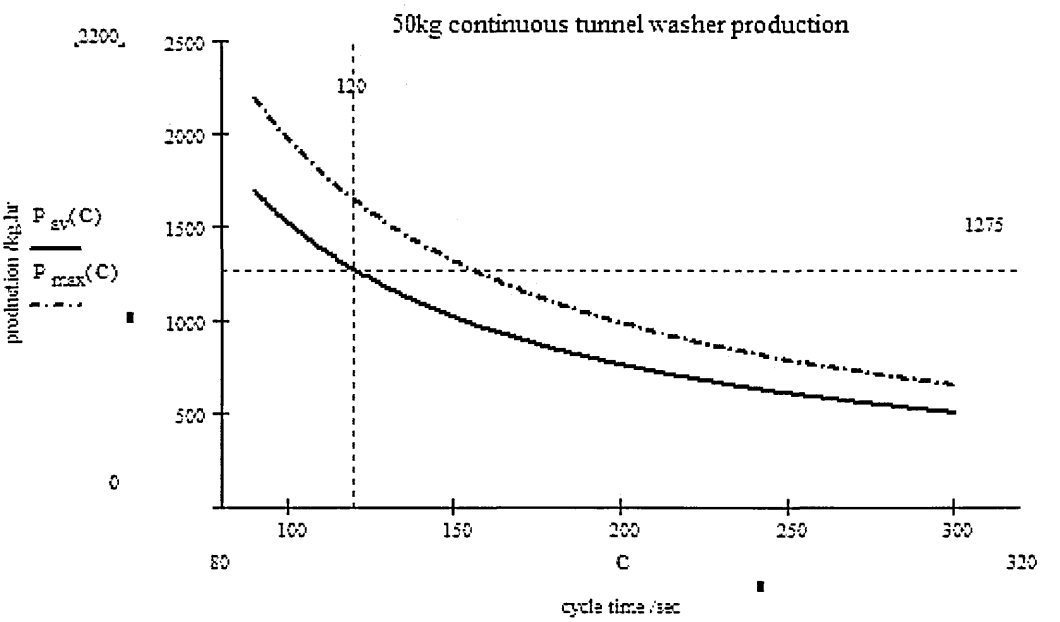


Figure 10: Continuous tunnel washer production

Figure 10 shows the variation of production with cycle time, for a 50kg continuous tunnel washer. Cycle time varies from 90sec (the minimum possible due to the press cycle) to 300sec (maximum realistic). Most industrial installations operate at between 90sec and 150sec. Note that in theoretical terms, the production is independent of the number of compartments. However, given that in reality the overall wash time (T_{wash}) has to be between 22-35 minutes, then this is a practical constraint.

After washing & drying, the work will be finished, according to its category. For example, towels will have been fully dried and need only be folded. Sheets and other 'flat' linen will be left damp and ironed. Each of these processes involves a high degree of automation. Again, it is necessary to have the work separated into

types, in order that it can be dried for the correct time and then conveyed to the appropriate finishing department.

The method of calculation presented above is presented in a more rigorous way than previously. There is no accepted method of calculating washer capacity (each manufacturer tending to have their own house style, all being a variation - usually simpler - on the scheme presented above). For example see (Rogers, 2003) and (Beggs, 2006).

3.3.1 Worked Example of Specifying a Washer

The following is an example of the normal process of specifying the size and length of a tunnel washer:

- decide on amount of work per hour to produce
    - usually this is determined by the operator, for example, they may specify that the tunnel washer is for a factory that must produce 300,000 pieces in a week, operating 60 hours. Some analysis must be done to determine, for example, typical piece weight, and the operator will have some discretion on hours' work, and also may have some need to include capacity for future expansion, but ultimately a rate of work - for example, 2500kg per hour - must be determined.
- decide on best batch size
    - this is a subjective choice depending on factors such as what downstream equipment may be used, personal preference of the operator, and the customer base of the operator (for example, one operator may have many small customers, and would therefore

find it difficult to make up larger batches, and if operating a larger washer would significantly underload it on a regular basis, reducing efficiency). In this case, say, 75kg loads were chosen.

- 2500kg per hour / 75kg batch = 33.3 batches per hour.
- based on a typical utilisation factor of 85% the system should be sized to produce a maximum of 33.3 / 0.85 = 39 batches per hour (in order to actually output an average of the required production)
- To produce a required 39 batches per hour, the cycle time must be 3600 / 39 = 92s
- Decide on the required maximum wash time
    - this will be done in consultation with the operator and the operators' washing chemist - and will be subject to the types of work to be produced. In relation to hospital work there are formal guidelines[2] but otherwise 25 minutes would be typical

  25 x 60 seconds, total 1500 seconds / 92s per compartment, = 16.3 (round up to 17) compartments

- Therefore for this example, a 75kg batch size machine with 17 compartments, washing at 90-95s cycle time, would be chosen.

Note that it is not to be implied that there is no benefit of calculating capacity in this manner - merely that it effectively sets the upper limit of the system, with actual performance always some level below anything predicted by this calculation.

[2] example for the UK - NHS Health Service Guidance HSG(95)18 requires minimum 3 min at 71'C or above for thermal disinfection, and Health Technical Memorandum HTN - laundry design - suggests a guide of 27 minutes total in the washing machine

3.4 Calculation of Dryer Capacity

Figure 4 (section 1.1) shows in schematic a typical batch washer installation with two washers, and seven dryers.

The previous section gave a method of calculating the size of the washer or washers. Because an over-specified washing machine is more expensive (in all terms - capital cost, maintenance costs, and running costs), it is important to calculate the minimum size of machine that can produce the necessary work. Therefore it is important that this machine - once installed - is kept running constantly with minimal interruptions - otherwise production will be lost.

The dryers and other downstream machines must therefore be equally carefully specified in order that they have capacity to handle all the work produced by the washing machines at their design speed.

A key parameter is the cycle time (C) and therefore the number of loads per hour that will be produced by the washers, and which then must be handled by the dryers.

As with the washer however, to specify the system with more than the minimum number of dryers, would cost unnecessary capital (a typical dryer costs approximately £80,000 and so to have unnecessary extra dryers on a system is very expensive).

A typical method of calculating the dryers is as follows:

(from the batch washer sizing)

C cycle time (seconds)

U utilisation factor (approximate actual production as a % of maximum)

If C=120 seconds, and U=85% then for a two batch washer system as shown in

Figure 4, the dryer group would have to deal with a range of 51-60 loads per hour. For dryer calculations, the maximum is usually taken. In this way the dryers should not hold up the washers.

It is important to know *a priori* the mix of work in terms of the length of drying time required. For example - a typical mix of work would be:

- 30% to be fully dried, with a total dry cycle time of 16 minutes
- 30% to be partially dried, with a total dry cycle time of 6 minutes
- 40% to be broken up in the dryer only, with a total dry cycle time of 1 minute

(the drying requirement depends on the next operation for the linen type, for example, some articles are ironed where they are required to be damp, while some articles are simply folded and packed, and required to be fully dry. Equally, different fabric types require different drying times. The moisture retention following drying is carefully controlled).

In the example being followed, where the dryer group has to deal with 60 loads per hour:

- 30% fully dried = 18 loads per hour
- 30% partially dried = 18 loads per hour
- 40% broken up = 24 loads per hour

Each load requires a number of minutes in a dryer:

- 30% fully dried = 18 loads / hr x 16 min = 288 dryer-min / hr
- 30% partially dried = 18 loads / hr x 6 min = 108 dryer-min / hr
- 40% broken up = 24 loads / hr x 1 min = 24 dryer-min / hr
- total 420 dryer-min / hr

Each dryer provides 60 dryer-min / hr, therefore the total number of dryers required is

- 420 dryer-min / hr / 60 dryer-min / hr = 7.0 dryers

The above analysis is expressed more concisely as the dryer system static model – shown in Figure 11 below.

Dryer system static model

washer cycle time /s	$Cycle := 120$
load size (batch size) /kg	$L := 50$
maximum production per washer	$P_{lds} := \frac{3600}{Cycle}$ *
number of washers	$N_{wash} := 2$
production total	$P_{total} := P_{lds} \cdot N_{wash}$ *
	$P_{total} = 60$ total loads per hour
breakdown of work into (3) categories	$B := (30\%\ \ 30\%\ \ 40\%)$
dry times for categories	$D := \begin{bmatrix} 16 \\ 6 \\ 1 \end{bmatrix}$
Dryers required	$N := \frac{P_{total} \cdot B \cdot D}{60}$ *
	$N = 7$

Figure 11: Dryer system static model

In practice it is rare for the required number of dryers to be an integer, and a judgment has to be taken - usually rounding the number up. Of course another

constraint might be the space available and it physically may not be possible to install more dryers.

In the schematic shown in Figure 4, the unloading conveyor should also be checked. It should have capacity to deal with at least the total required number of loads per hour, and preferably a significant overcapacity to deal with the surge inevitably experienced when several dryers need to unload at a similar time. In a system such as this, the dryers will be interlocked so that they cannot unload at the same time (which would cause loads mixing up on the unloading conveyor). The dryer checks that the unloading conveyor is free, and unloads. The conveyor control registers that it has a load on, and prevents further dryers unloading until it has discharged the load. Clearly a dryer finishing its drying cycle while the conveyor is full would simply have to wait, and this adds very significantly to the required capacity of the dryers.

Also note that this entire calculation is predicated upon the assumption that the work coming through the washers is mixed evenly with regard the stated dry codes, that is;

- 30% fully dried
- 30% partially dried
- 40% broken up

If this is not the case then the entire washline can easily be held up which causes cost and logistical problems to the plant.

3.5 Conclusions Relevant to Research Objectives

It was concluded that variation of work in both short and long term is commonplace. Furthermore variation is not often predictable, and most laundry operators do not observe the variation at the time or even afterwards. Therefore there is a commercial benefit to improving the self-optimisation of the processes.

Over or under specifying a wash system is detrimental to efficiency in operation and may also lead to wasted investment or insufficient capacity. Therefore there are advantages – commercially and for sustainability – to improving the manner of simulating linked sequential systems in industrial laundries, and hence predicting their performance.

Chapter Four: An EANN Unsupervised Classification System

4.1 Introduction

This chapter describes an early attempt to implement an EANN to produce a classification decision without using traditional Supervised Learning. In this case, the only *a priori* information about the objects to be classified was the relative proportion between the two classes.

The main details of this work have been previously published (Morley, 2005).

The classification problem in industrial laundry is important, as work presented to the laundry is usually presented mixed (towels, sheets etc) and these must be separated in order to be washed and further processed. It was considered that as a first investigation into the field of industrial laundry, and following on from undergraduate work, EANNs could be applied to the problem of classifying laundry pieces, as the industry requires a solution which has minimal skilled human intervention, and the relative proportions of different pieces is known. The normal method of classification currently in use, is manual with unskilled operators classifying items by sight. This method is slow, relatively expensive, and subject to approximately a 5% error rate. (Morley, 2003).

The underlying premise is that given a population of two types of object, where the relative proportions of the two objects are known, the proportion of types can be used to classify the objects. For example,

Figure 12 below shows a sample of a population where it is known that 30% of the objects in the population are type A, and 70% are type B.

Figure 12: Population of unknown objects

There are 64 of the smaller object, and 28 of the larger object shown in

Figure 12. They therefore split approximately in a 70:30 ratio. From this evidence it would be reasonable to conclude that the object shown in Figure 13 is type A.

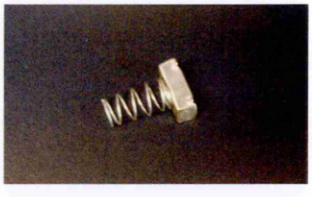


Figure 13: Larger of the two objects

However there will still be a level of uncertainty about this, and this uncertainty will reduce as the sample viewed becomes larger and larger.

The hypothesis in this research was that this same type of *a priori* information could be used in evolving an ANN classification system.

A method of unsupervised learning was proposed that uses an EA to train an ANN to implement a classification function using the relative proportions of the subject to be classified. The EA was used to evolve the weights in a network in order to provide some classification function that provides the same proportional split as that known, and it was considered that this would provide the classification function required.

It was furthermore proposed that with a minimal amount of human guidance the performance of the evolution can be directed, in an analogous manner to a human dog breeder who utilises the mechanics of evolution to a conscious end. However this avenue of further work was not followed up, but offers scope for the future. For example, an operator would not be available to guide the evolution of the network normally, but maybe could occasionally - after say an extended period of autonomous evolution - select or reject a number of networks.

4.2 Experimental Method and Results

For simplicity, an ANN was implemented on a spreadsheet programme with the workings of the network realised as formulae within the spreadsheet. The EA was implemented using VisualBASIC routines embedded in the spreadsheet. This made the system very simple to program and apply. It was able to be implemented and the experiments run on a normal office computer with no special software or requirements.

4.2.1 General Description

The approach described above, was applied to the classification of images. The images were very coarse - 12 x 12 grayscale pixels - and the images presented were either a circle or a cross, with the possibility of some form of degradation applied.

A population of 20 ANNs was used, initially generated with random weights.

The images were presented in turn, to each ANN in turn. Each ANN had a single output which classified the presented image as either type A or B. A count was made of the number classified as the different types and the proportional split calculated.

After presenting the set of images to the ANNs in turn, the ANNs were placed in rank order, according to how their output proportions matched the known class proportion.

Furthermore, after the EA has found a number of different networks which do indeed classify in a 60:40 split, an operator could then present several images (in any proportion) to each of the networks, and reject those which are not effective in classifying images as required. This extra selection can be seen as ensuring that the evolution stays on the required track. It is justified by the observation that to manually classify a reasonably large training set of images will be time consuming. However, to look at several images which should all be apples, and notice that a large number of them are bananas, is a quick human operation.

4.2.2 Implementation

The network was structured as follows

- Input layer, 14 neural units each giving a weighted sum of 5 pixels
- First Hidden layer, 5 units with 14 weighted inputs, giving a weighted sum of every one of the input layer units (full interconnection)
- Second Hidden layer, 5 units with 5 weighted inputs
- Third Hidden layer, 5 units with 5 weighted inputs
- Output layer, 1 unit with 5 weighted inputs

This is shown in Figure 14,

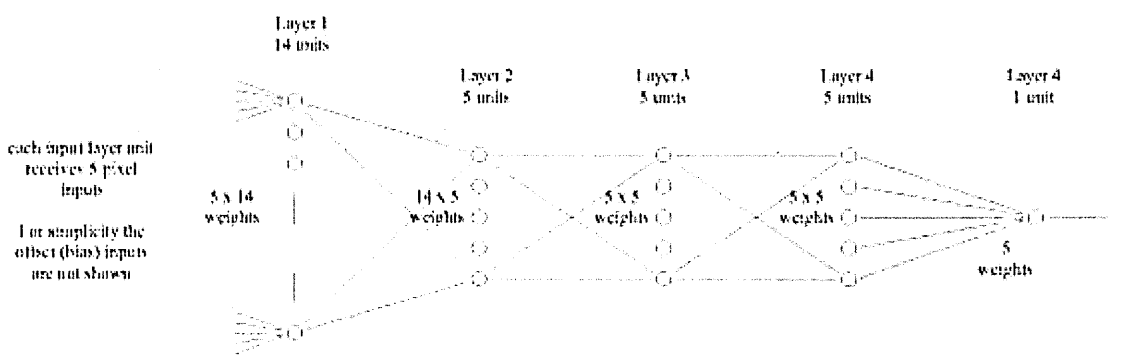


Figure 14: Network structure

A classification function arbitrarily classified the input image as either 'type A' or 'type B' according to whether the output of the output layer was positive or negative.

This multi-layered network architecture is different from the normal architecture, which has one hidden layer (see Figure 5 from section 2.1.2). The normal architecture is commonly used and well documented in the literature. For this early experiment it was felt worth investigating a form of network which was

different. The number of units was chosen arbitrarily, but intuitively felt to be enough that there was a large search space (the space encompassing all possible link weights) within which a sufficiently good approximate solution could reasonably be expected to be found.

Simple images were presented to the network, each being 12 x 12 grayscale pixels. Figure 15 below shows two such images - the remainder of the images were taken from these two images but following processing by the addition of noise, by an operation on the brightness, or by translation.

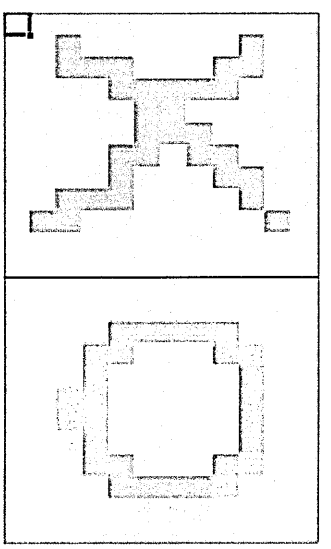


Figure 15: Seed images

Following work by (Johnson & Rose, 2005) which itself drew upon Simon's Three Pixel Principle, it is not essential to view - take as input - every single pixel. In this experiment 70 out of the total 144 pixels were chosen as inputs into the network. The Three Pixel Principle states that a feature of an image will show in at least three pixels, (or not at all). Therefore, taking as input fewer than 100% of the pixels will substantially reduce the computing power required, but not affect the result.

Each ANN was described by a chromosome - a sequence of 140 integers (the x & y co-ordinates of the 70 pixels taken as inputs) and then 195 floating point numbers - the weights of the neuron links.

Twenty such networks were created with random sequences of numbers (being the link weights and also the pixel coordinates). For each network in turn, 50 images were presented and the proportions that that (random) network split the images into were logged.

The 50 images were generated from two 'seed' images according to a probability function p(A), and the copies were processed by adding random noise, modification of the brightness or contrast, or translation 1 pixel in a random direction. Therefore although 48 images were all originally copied from the same 2 seeds, no two of the 50 images were quite the same.

The EA then applied was;

- Evaluate each network according to the fitness function which simply ranked the 20 networks according to which had achieved a proportional split of the images closest to that known a priori as the proportional split between types A & B.
- Eliminate (delete) the weakest (least fit) 10 network instances.
- Generate a new 10 network instances by picking pairs of the remaining (most fit) networks at random, picking a random split point, and then combining the top half of one network with the bottom half of the other.
- Perform a mutation operation where a percentage of every number in every member of this new generation is subject to a random change.

The new generation of network instances was then presented to the same 50 images and the whole process repeated.

4.2.3 Series 1 Experiments - Results

The following parameters could be set for each network series;

- number of generations to run
- mutation rate
- the proportion split of the images (p(A) is the probability of a sample being type A)
- the variation or degradation method of the images

Firstly, some experiments were performed in order to get some idea of what mutation rate to use, and how many generations it typically took to home in on a working solution.

4.2.4 Number of Generations

Performing these runs for 10 and 20 generations, with different combinations of the above parameters, gave the results shown in Figure 16.

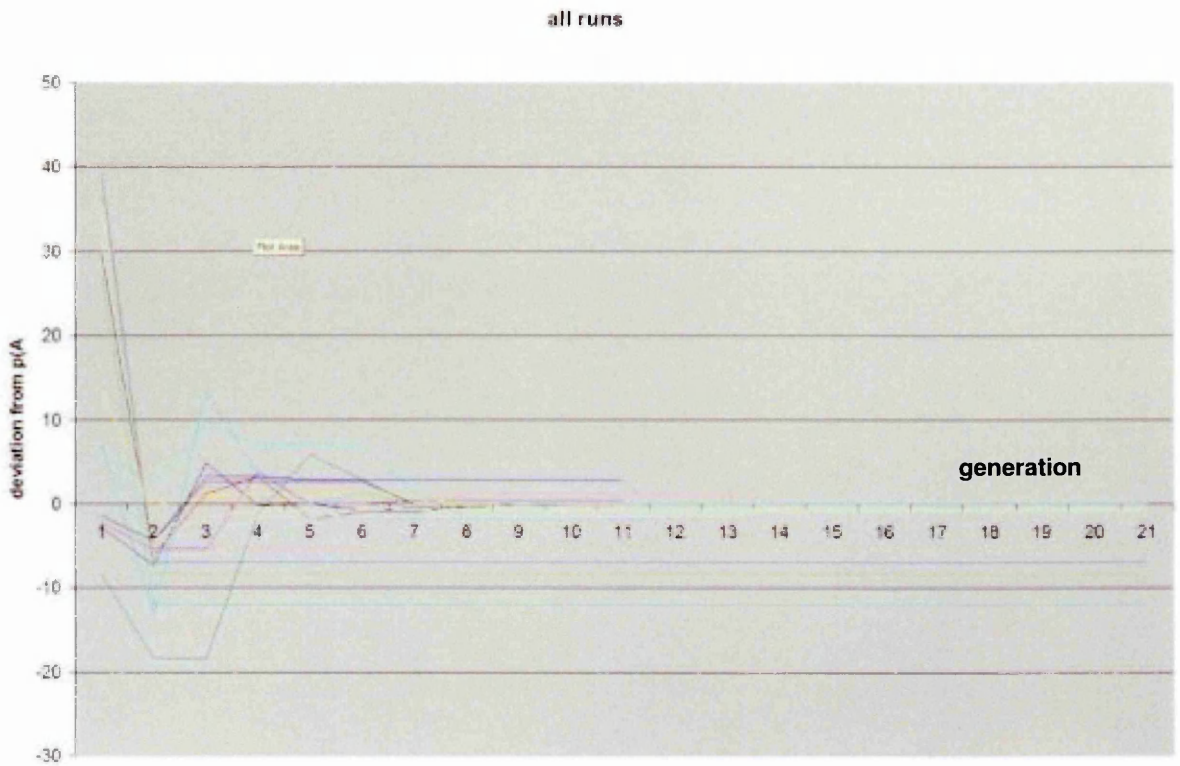


Figure 16: Results for initial experiments

In each case, according to the EA procedure described above, after evaluation, the 10 better network instances were used for recombination. The graph shows the mean of deviations from p(A) for these 10 better networks (a measure of the overall performance of the population of networks), with generations along the x-axis, each trace representing a different experiment with different parameters. Therefore as the evolution proceeds the traces tend towards 0 on the y-axis (i.e. no deviation).

Figure 16 shows empirically that in this context, most network generations home in on an effective solution within 8 generations, with little further improvement seen in the following 12 generations. Therefore it was decided to perform all future experiments with 10 generations. This was determined in the early part of

the experiment. Therefore further experiments were run for 10 generations only. These are the traces which stop halfway along the graph.

More detailed explanation of the individual experiments follows.

4.2.5 Mutation Factor

Some of the runs were performed with different mutation factors, holding other parameters constant. Results for one such collection of runs are shown in Figure 17 below. This is an expansion of some of the traces from Figure 16. The label for each trace shows both p(A) and the mutation factor. For example: trace 5, 40-5 means p(A) is 40, and mutation is 5%.

Here, 5 runs were performed with the mutation factors 1-5, with P(A) approximately held at 40%. Along with other similar trials, there is no definitive 'best' factor to use, but as the traces with 3-5% mutation factor gave reasonable performance. Further work could be beneficial in determining to a greater extent the optimum number of generations, and mutation factors but it was considered that a more detailed study of this would not benefit this research work at this stage.

For the purposes of these experiments, it was decided to proceed using the figures of 10 generations & 5% mutation.

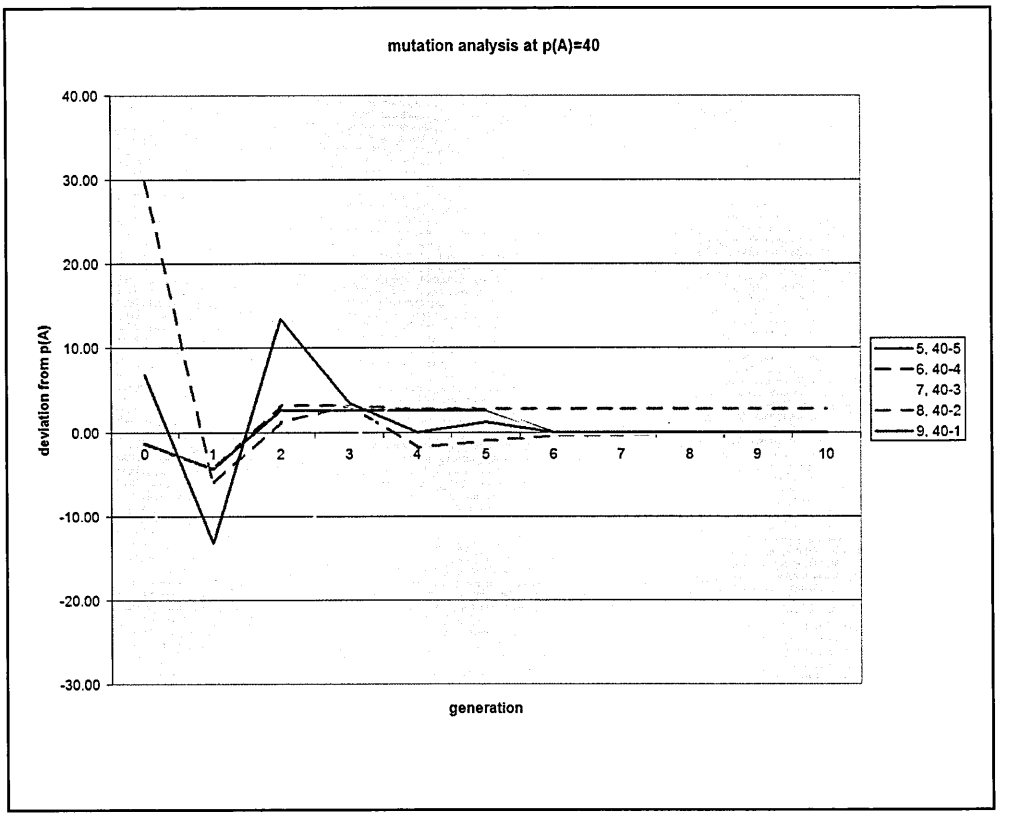


Figure 17: 10 generations with different mutation factors

3. 2.6 Series 2 Experiments

Here a similar process was carried out, but human interaction took place after 10 generations.

step 1

50 images were again populated from 2 seeds, with probability distribution p(A)=36%, and then processed by adding noise

10 generations of the EA were carried out as described above, selecting according to the effectiveness of the network in splitting the images into a proportion close to p(A). This created 10 networks, shown in Table 3Table 3 where for each network, the % given in the second column represents the split the network achieved in classifying the images. It can be seen that these compare favourably with the 36:64 actual split of the images.

network	% split achieved	errors
1	34%	0
2	40%	1
3	36%	1
4	36%	13
5	32%	7
6	36%	0
7	36%	2
8	34%	10
9	40%	2
10	40%	2

Table 3: Results for 10 networks with P(A)=36%

An operator then presented a different set of 50 images to the network (this time degraded by processing with a noise factor 5% where every value had a 5% chance of being changed to a random value, and also modifying the contrast of the images with a blanket factor of 5%) and observed the classification results. The numbers of classification errors found are shown in the third column of Table

3. The networks 4, 5, and 8, which gave the highest number of errors, were eliminated, and the networks 1 & 6 were copied into their places (with network 6 being copied twice), thus eliminating networks giving worst performance, and rewarding (by reproducing) networks giving best performance.

(Curiously, note that network 4, which gave a perfect 36% split on the initial set of images, performed worst during the human interaction despite only slight differences in the set of images)

Step 2

The process was repeated with this revised set of networks, this time starting with P(A)=18%, and using images processed with 10% noise and 5% brightness. The results are shown in Table 4. It can be seen that after this selection all networks give a perfect 18% split, and far fewer errors were found during the human interaction stage using a different set of test images (this time with 30% noise and -10% contrast).

Networks 1, 2, and 4 were manually eliminated, 2 being the worse performing network (most errors) and 1 & 2 being chosen randomly from the set of all networks giving some errors. They were replaced with copies of networks 3, 6, and 7, these chosen randomly from the set of all

network	% split achieved	errors
1	18%	1
2	18%	2
3	18%	0
4	18%	1
5	18%	1
6	18%	0
7	18%	0
8	18%	0
9	18%	1
10	18%	0

Table 4: Experiment results with P(A)=18%

networks giving no errors.

Step 3

The process was again repeated with this revised set of networks, this time starting with P(A)=36%, and using images processed with 10% noise and a random translation in any direction. The results are shown in Table 5. It can be seen that after this selection the network's effectiveness is reduced - translations being harder to classify. The numbers of errors shown during the human interaction stage are also far greater than previously. The set of images used at this point comprised translated images coupled with 10% noise factor as before.

network	% split achieved	errors
1	28%	17
2	32%	5
3	28%	17
4	32%	4
5	32%	5
6	32%	5
7	36%	6
8	32%	21
9	32%	22
10	36%	22

Table 5: Results with P(A)=36% and noise added

Step 4

Networks 1, 3, 8, 9, 10 were eliminated- these having the most errors as shown in Table 5, and replaced with copies of 2, 4, 5, 6 Network 4 was copied twice as it had the fewest errors. The process was again repeated with this revised set of networks, this time starting with P(A)=30%, and using images processed with 10% noise and a random translation in any direction. The results are shown in Table 6. It can be seen that after this the network's effectiveness is improved from the last set, given that this is selecting from a set of translated images. The number of errors shown following the human interaction using a new set of images is also improved. (Once again comprising translated images coupled with 10% noise factor).

network	% split achieved	errors
1	20%	4
2	20%	4
3	20%	4
4	36%	9
5	20%	4
6	20%	4
7	32%	11
8	36%	9
9	20%	4
10	20%	4

Table 6: Results with P(A)=30% and noise added

Step 5

Networks 4, 7, 8 were eliminated, and replaced with copies of 1, 2, 3.

The process was again repeated with this revised set of networks, with P(A)=20%, and using images processed with 10% noise and a random translation in any direction. The results are shown in Table 7, and here the network's effectiveness is again improved. The test set of images used during the human interaction stage again comprised translated images coupled with 10% noise factor.

The performance consistency of the networks is due largely due to the fact that at this point the cross-generation of the successful networks has led to the proliferation of the successful weights across all networks, meaning that the 10 different networks are virtually the same.

network	% split achieved	errors
1	20%	4
2	20%	4
3	20%	4
4	20%	4
5	20%	4
6	20%	4
7	20%	4
8	20%	4
9	20%	4
10	20%	4

Table 7: Results with P(A)=20% and noise added

4.2.5 Series 3 Experiments

A similar process was carried out as described for series 2. However in this case all image processing was restricted to random direction translations (typically the hardest type of image modification to recognise) coupled with noise addition.

'Natural' evolution was carried out for 10 generations, followed by artificial selection (rejection of the worst performing networks). All natural and artificial selection processes were carried out with different sets of images.

After 4 such cycles, the resultant 10 networks offered perfect classification over 5 different sets of images with random translation, and also between 92-94% correct classification for sets of images with random translations coupled with 25% random noise addition (at which point the images were noticeably degraded but still possible for a human to fairly easily identify them).

Figure 15 previously showed the two seed images. Figure 18 shows the same images with a random translation and then the addition of 25% noise.

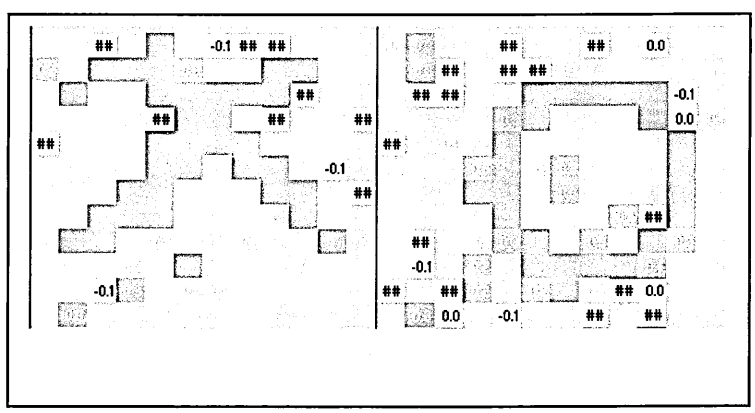


Figure 18: Seed images with noise added

Noise was added according to the following formula: if the noise factor is 25% then every pixel is given a 25% probability that it will be changed to a random value.

Over the course of this experiment the successful networks bred and therefore became copied across the family, and one network came to dominate with 7 out of 10 instances (c.f. Schema Theorem, section 2.2.3). The retinal pattern - pixels used as inputs - for this successful network is shown in Figure 19. Although there are 70 network inputs, there are only 58 pixels shown as inputs. This is because there was no mechanism to prevent the same pixels being used as multiple input, and in this case, the network has evolved a pattern where 12 pixels are used as more than one network input.

The retinal pattern shown has a cross to represent a pixel which is used as an input to the ANN. Roughly 50% of the image pixels are used as an input, and roughly 50% are ignored.

The retinal patterns are evolved, with the same mechanism as the network weights, and it is intuitively presumed that over evolutionary time the retinal pattern would optimise itself for the context, i.e. the particular images presented.

In this case, the retinal pattern presented does not indicate any particular form from which any further conclusions can be drawn, and it is included merely for completeness.

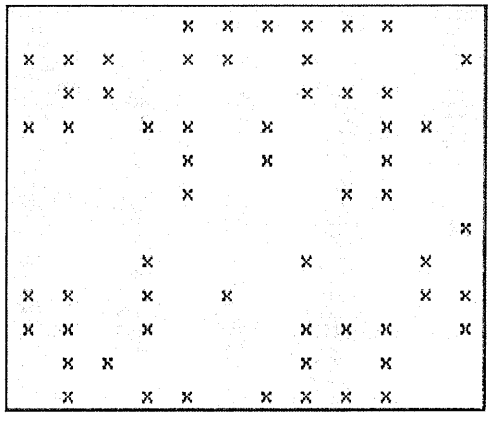


Figure 19: Retina pattern for successful network

4.3 Conclusions & Discussion

These experiments show empirically and qualitatively that the use of an EA can be effective in generating an ANN to approximate a classification function, which distinguishes between classes, where a training set of pre-classified samples were not available, and the only information given *a priori* is a known proportional split between samples.

The experiments indicate that the EA homes in on a reasonable solution within 10 generations, although this is a highly contextual result. It is not considered that there is any reason to suspect that this would hold more widely.

The longer experimental series furthermore indicate that manual intervention in-between sequences of EA breeding can enhance performance. This is broadly analogous to the artificial selection used by domestic animal breeders, as opposed to natural selection (normal evolution) set within a context environment. The dog breeder needs no knowledge of how genetics works, but can over time breed dogs with ever longer tails. Here, a human operator needs no knowledge of how the algorithm works, but by throwing out networks which are diverging from the required, and retaining those which appear to be successful in the task, the evolution can be influenced. It is also feasible that this may be automated.

This set of experiments investigated a new direction in combining ANNs and EAs, and it is believed to offer a powerful new approach to many classes of problems.

In relation to the initial scope of this chapter, the approach was an initial set of experiments. The experiments here indicate that the EANN approach could be

used successfully (in combination with suitable mechanical equipment to separate the laundry pieces and image capture equipment to provide an input to the EANN system) to classify laundry pieces. However during these experiments it was realised that there was another potential application for this methodology: the control of conveyor systems, where a pattern of input data is used to generate a single output control decision. This relates directly to Research Question 1.

In relation to Research Question 1 (section 1.2) these experiments showed in general and qualitative terms that the EANN strategy proposed can be used to effectively make a decision based on input data, and the EA can generate ANNs that are effective in a decision making task when trained only with data representing the proportion of type A / type B decisions required. This implies that this strategy can be used in control of Linked Sequential Systems in Industrial Laundries of the type described in chapter 3.

This is a very early stage in the development, and several questions for further work are immediately apparent;

- How does this form of hybrid perform in a wider context with more complex inputs, more variations, and more networks breeding? Would the extra computation involved render this technique nice but ultimately impractical?
- Will the artificial selection procedures be actually necessary? What will be the conditions which would prove that this is beneficial? Is there an optimum level of artificial/natural selection or will this vary from context to context?

- Will the behaviour of a network be something that can be predicted or will it ever only be emergent? With this technique there is a large random element and so any behaviour modelling will have to be based on statistical probability methods; what (if any) accuracy will there be?
- In what contexts - if any - will this technique prove useful in preference to other established AI or conventional techniques?

However, this further work was not pursued in this research as it was necessary to focus on the specific research questions.

Chapter Five: EANNs for Industrial Control

5.1 Introduction

This chapter describes an experiment into the use of an EANN (similar in principle to that outlined in chapter 4) for controlling a part of a Linked Sequential System in an Industrial Laundry - a conveyor system. A single ANN was used as a black-box decision maker about the next operation to be executed, and an EA was used to find the link weights. Part of this work has been previously published. (Morley, 2010)

This work is directly relevant to Research Question 1, as it investigates an EANN to the control of linked sequential systems in industrial laundries, in particular at a key decision making point. Here the linked sequential system is normally controlled with a conventional PLC control which (for the reasons outlined in section 3.2) is normally operating sub-optimally. The EANN approach offers scope for continuous adaptation to changing circumstances.

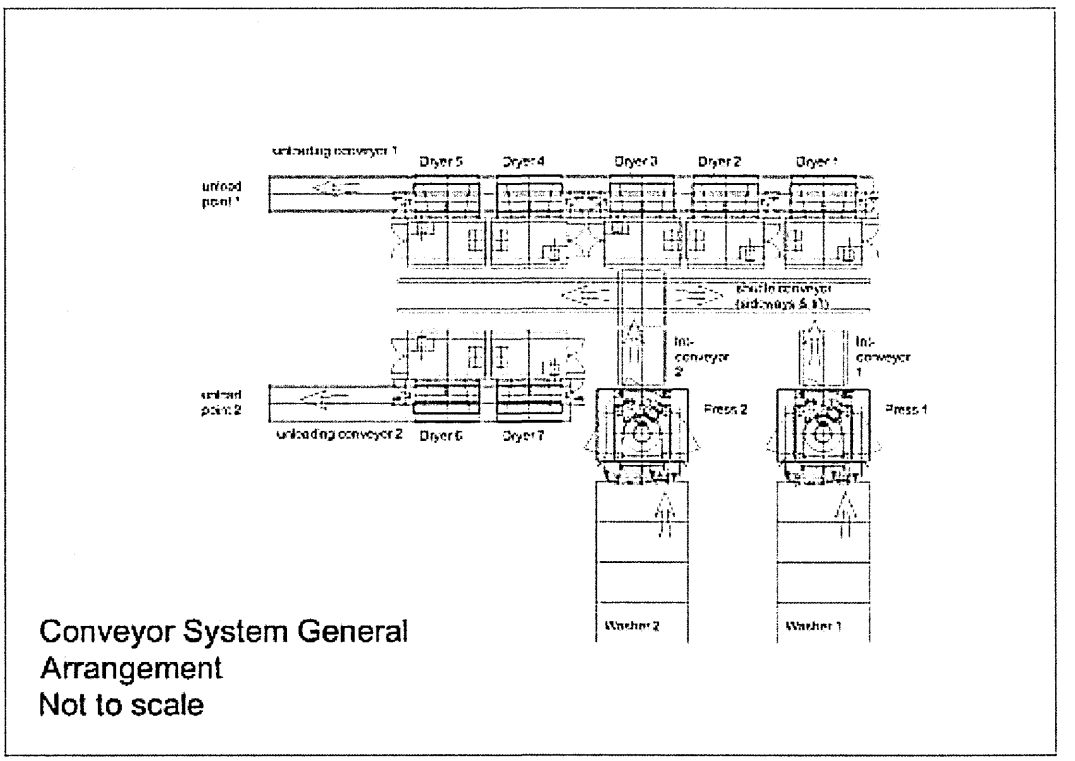


Figure 20: Washing system general layout

Figure 20 shows the general layout of the system. This washing and drying system comprises two batch washing machines and seven dryers, and was installed in a commercial laundry in Ireland in 2005.

The batch washers are followed by a press which extracts excess water. The pressed batches are then stored on a short conveyor which forms a buffer between the press and the shuttle conveyor. The single shuttle conveyor moves sideways on a track collecting work from the two storage conveyors, and transferring it to one of the seven dryers. Finally the dryers unload into one of two unloading conveyors which send the dried work on to downstream processing.

The washers operate on a 2 minute[3] cycle time and therefore each can produce a maximum of 30 loads per hour (60 loads per hour total). It is commercially important that as much as possible of this theoretical maximum production is actually produced. Otherwise further processing machines downstream of the wash-dry system will be idle and the consequences of this will be both costly in terms of wasted resources, and may potentially lead to shortages to the hospitals being supplied.

The bank of 7 dryers forms a bottleneck, and a key performance requirement is the optimisation of production through these dryers. If for example, the dryers are all full, then the shuttle has nowhere to take batches of work produced and the washers have to stop until a dryer becomes empty.

There are lots of different types of laundry work. However, the main relevant factor is the drying requirement and, in the design of this system, the three categories given in section 3.4 were used: these being no-dry, part-dry, and full-dry.

The system was designed assuming a work mix[4] of:

- 52% no dry
- 18% part dry
- 30% full dry

---

[3] In fact the cycle time was designed to be 130 seconds, but the system was commissioned and operated at 120 seconds from the start and this proved sufficient.

[4] The system was actually designed for two different work mixes in two different shifts. The work mix given here is a simplification, as are the dry times, which are given as an average over several different times for different programmes.

The design calculation of number of dryers required is given in appendix D, and follows the process outlined in section 3.4. It predicts the number of dryers required as 6.7, and hence 7 were installed[5].

It can be seen that there is very little surplus drying capacity. Hence the dryers form a bottleneck and their efficient utilisation is critical. Clearly the full dry work uses the majority of the dryer capacity. Should work mix vary from the design assumption, with more full dry work than planned then the dryer capacity would no longer be sufficient.

The performance is affected by the transit time of the shuttle. For example, if the shuttle is in position for dryers 5 & 6, and a load of linen becomes available from washer 1, then washer 1 cannot unload until the shuttle has moved to that position, thus losing that time for the washer.

If a shuttle puts long-drying work into a dryer close to the washers, then it ties up that dryer for a long time. Hence dryers further away have to be used leading to longer transit times.

More subtle effects still cause sub-optimal performance, such as the tie-up of the dryer unload conveyors caused by loads from, for example, dryers 1, 2 & 3 holding up the unloading of dryers 4 & 5.

Optimal utilization would achieve 30 batches from each washer per hour. The mix of work (with different proportions of batches with different dry times) varies over time and this causes a variation of performance.

---

[5] Space was also a limitation. There was insufficient space for more dryers in the factory.

5.1.1 Conventional Control

See Figure 20 for the general system layout. The system is currently controlled by a conventional control system with simplified pseudo code algorithm as follows:

i. IF shuttle is empty AND intermediate-conveyor1 is full, THEN
    - move to position 1 and load shuttle

ii. IF shuttle is empty AND intermediate-conveyor2 is full, THEN
    - move to position 2 and load shuttle

iii. IF shuttle is empty AND both intermediate-conveyors are empty THEN
    - move to position 1

iv. IF shuttle is full AND not all dryers are full THEN
    - move to first available dryer and unload

v. IF shuttle is full AND all dryers are full THEN
    - wait

It can be seen that this algorithm is simple, therefore easy to implement, easy to commission, and easy to fault-find. It is also sub-optimal.

For example, with respect to step v, it would be better to move to the dryer which will become free next, thus saving the travel time when that dryer does become free.

Similarly, with respect step iv, a better strategy where a choice of dryers is available would be to put loads requiring a longer drying time in a dryer which is further from the washers (thus reducing overall shuttle travel time).

An ideal control system would also equalise use of the dryers leading to more even wear and tear.[6]

Conventional control typically achieves 85-90% of optimum performance but over time this will decrease for the reasons outlined in section 3.2

5.2 Experimental Method

The system was simulated in a bespoke VisualBASIC program run on a normal office PC.

5.2.1 EANN Strategy & Parameters

A population of 50 ANNs was generated with randomised weights, each with an input layer of 28 units, and a hidden layer of 13 units, and then an output layer with 9 units. As the output was required to be a number between 1-9, a winner-takes-all strategy was implemented. The network output, was the number of the output layer unit with the highest output value. Each ANN was a fully-interconnected feed-forward network of the general form shown in Figure 21.

[6] Otherwise it is an implicit assumption that the performance of all dryers is identical, and this is not necessarily the case.

There were 28 x 28 layer 2 weights, 28 x 13 layer 3 weights, and 13 x 9 weights for the output units. In total this was 1265 weights per network.

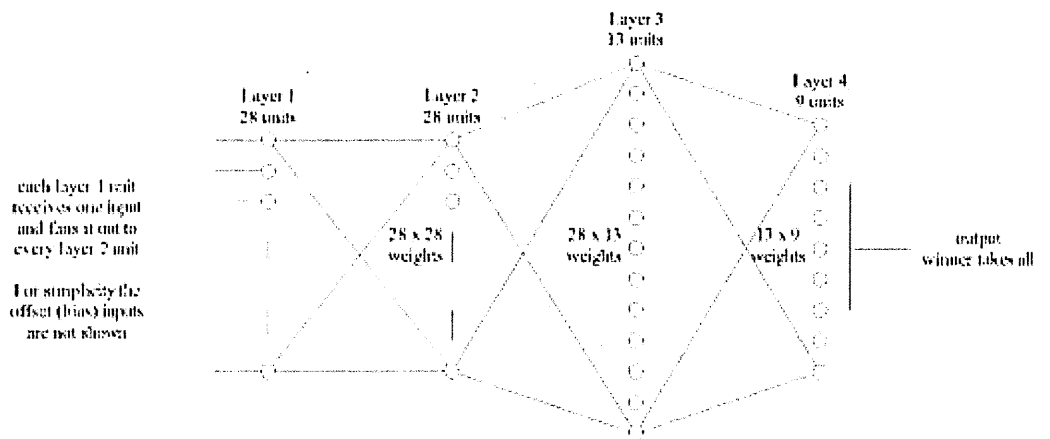


Figure 21: structure of each ANN

The inputs to the ANN were parameters representing the state of the system. For example, a number representing the current load in the press, on the conveyor etc.

Each ANN was represented as a sequence of its link weights, and these sequences were the 'chromosomes' to which the EA was applied.

Each unit implemented a weighted sum of all its inputs ($\sum_i w_i x_i$) and then standard sigmoid function as a conventional neural network implementation.

$$\text{unit output} = \frac{1}{1+e^{-\left(\sum_i w_i x_i\right)}}$$

This function (described in (Picton, 1994)) is chosen so that the output from each unit is non-linear, and differentiable. In fact there is no need for a differentiable

function if BP is not implemented. However, it is relatively easy to implement and leads to a network that is consistent with conventionally understood ANNs.

In total there were 1265 weights, comprising;

$$(28 \times 28) + (28 \times 13) + (13 \times 9)$$

The inputs to the ANN were constructed as an array of originally 26 values. The shuttle load-on and current position were then appended to make it a 28 value local array. The inputs were mapped as shown in Table 8. Most inputs represent the type of batch currently in that position. For example, for position 1 – load (currently) in the press, a '0' means the press is empty, a '1' means a batch with dry code 1 is currently in the press, a '2' a batch with dry code '2' and so on.

index	description		range
0	load on intermediate	wash-line 1	0-3 (0 indicates empty)
1	load in press		0-3 (0 indicates empty)
2	load in last compartment		1-3
3	load in penultimate		1-3
4	time remaining		0-120
5	load on intermediate	wash-line 2	0-3 (0 indicates empty)
6	load in press		0-3 (0 indicates empty)
7	load in last compartment		1-3
8	load in penultimate		1-3
9	time remaining		0-120
10	load currently in	dryer 1	0-3 (0 indicates empty)
11	time remaining		0-960
12	load currently in	dryer 2	0-3 (0 indicates empty)
13	time remaining		0-960
14	load currently in	dryer 3	0-3 (0 indicates empty)
15	time remaining		0-960
16	load currently in	dryer 4	0-3 (0 indicates empty)
17	time remaining		0-960
18	load currently in	dryer 5	0-3 (0 indicates empty)
19	time remaining		0-960
20	load currently in	dryer 6	0-3 (0 indicates empty)
21	time remaining		0-960
22	load currently in	dryer 7	0-3 (0 indicates empty)
23	time remaining		0-960
24	time remaining	unload conveyor 1	0-30 (0 indicates empty)
25	time remaining	unload conveyor 2	0-30 (0 indicates empty)
26	current shuttle position		1-9
27	current shuttle load on		0-3 (0 indicates empty)

Table 8: ANN inputs

Essentially the neural network in current control of the system is treated as a decision-making black box. The shuttle conveyor would carry out each operation autonomously and then ask the network 'where do I go next', and the network would give a command such as 'go to dryer x and unload'.

The EA aspect of the experiment proceeded as follows;

- create initial population of 50 random instances (the 'chromosome' was simply a sequence of 1265 floating point numbers representing all the weights for an ANN instance in a particular fixed order. The initial population was generated with every weight set randomly.
- run each network in turn and evaluate the performance of each based simply on a measure of how many batches were produced. (high level metric).
- always keep the best performing network.
- for every other network, if its performance was lower than the average performance, then remove it from the population with a 50% chance.
- for networks removed, generate replacements by:
    - select two parent instances at random from the set of networks that perform better than average.
    - take a random split point and recombine into a new chromosome.
    - pick a small number of weights at random and change them for random values (a mutation factor).

This is a relatively simple evolutionary strategy with the advantage of being simple to implement.

Note that this means that the best performing instances are retained through the generations without any modification (or mutation), and only the fitter individuals are selected as parents of the next generation.

Island Strategy

It was decided to implement an island strategy for the GA, because these have been shown to be a good way of preserving genetic diversity and allowing each island to search a different part of the solution space (Whitley, et al., 1998). This was carried over four phases, mutation parameters chosen based on previous work and designed to give a steadily declining mutation rate.

- *phase 1*

5 completely separate populations of 50 networks each were evolved in isolation, each over 100 generations. The mutation rate was set at 5% for two populations, 10% for two populations, and 15% for one.

- *phase 2*

runs 1 & 2: 100 generation, with 50 networks taken at random - 10 from each of the separate phase 1 populations.

run 3: 100 generations, with 50 networks, 10 taken from each of the separate phase 1 populations manually selected as better performing networks. For each the mutation rate was set at 5%.

- *phase 3*

With one population of 50 individuals, taking 25 from each of the two phase 2 populations. This was run for 400 generations with a mutation rate of 1%.

- *phase 4*

With one population of 50 individuals taken straight from phase 3. This was run for 500 generations with a mutation rate starting at 0.5% and steadily declining to 0.1% for the final 100 generations.

5.2.2 Simulated System Parameters

Simulated Work Mix

- 40% batch category 1 (for fully dry - 16 minutes - 960 seconds)
- 30% batch category 2 (for partial dry - 6 minutes - 360 seconds)
- 30% batch category 3 (for no dry - 1.5 minutes - 90 seconds)

The wash-loads were chosen in random order according to this probability distribution.

System Timings

The following times were chosen to be representative of real systems.

- loading the shuttle from the intermediate belt - 9 seconds
- unloading a shuttle into a dryer - 16 seconds
- unloading a dryer - 30 seconds - i.e. the run time for the unloading conveyor

Shuttle Travel Times

Table 9 shows the shuttle travel times from each position to each position. For simplicity this table was assumed symmetrical although in the field there is no reason to suppose this assumption holds.

	position								
	1	2	3	4	5	6	7	8	9
1 – washer 1	0								
2 – washer 2	14	0							
3 – dryer 1	8	22	0						
4 – dryer 2	15	15	7	0					
5 – dryer 3	22	8	14	6	0				
6 – dryer 4	29	15	21	14	7	0			
7 – dryer 5	35	21	27	21	14	6	0		
8 – dryer 6	35	21	27	21	14	6	0	0	
9 – dryer 7	29	15	21	14	7	0	6	6	0

Table 9: Shuttle travel times

5.3 Results

In each case, the number of batches produced was measured for each network. The following tables record the number of batches produced by the worst performing network, the best performing network, and the average of all 50 networks, for both the start point (i.e. with random networks) and for the population at the end of the 100 generations.

5.3.1 phase 1

(5 completely separate populations of 50 networks with 100 generations. Mutation rates of 5%, 10% and 15% for different populations.)

phase 1, 50 networks, 100 generations, 5400s simulated run time							
run	**mutation rate**	**initial**			**final**		
		worst	**average**	**best**	**worst**	**average**	**best**
p1 run1	5%	5	5.48	7	5	8.24	18
p1 run2	5%	5	5.68	9	5	6.88	11
p1 run3	10%	5	5.72	12	5	7.52	13
p1 run4	10%	5	5.72	12	5	7.52	13
p1 run5	15%	5	5.72	12	5	6.66	13

Table 10: Results, for phase 1

In Table 10 above, in each case the initial performance (recorded in batches produced by the system) was for ANNs with randomised weights.

In each case the average performance and the maximum performer steadily increased over the 100 generations. Figure 22 shows graphically the full results for phase 1 run 3, this being typical of each of the 5 experiments (the table shows only the starting and end points – the graph shows performance all the way through the generations).

While the average performance of the population increases steadily, in the control application it is the best performing ANN which is of particular interest, as it is the best performing ANN that would be used for actual control.

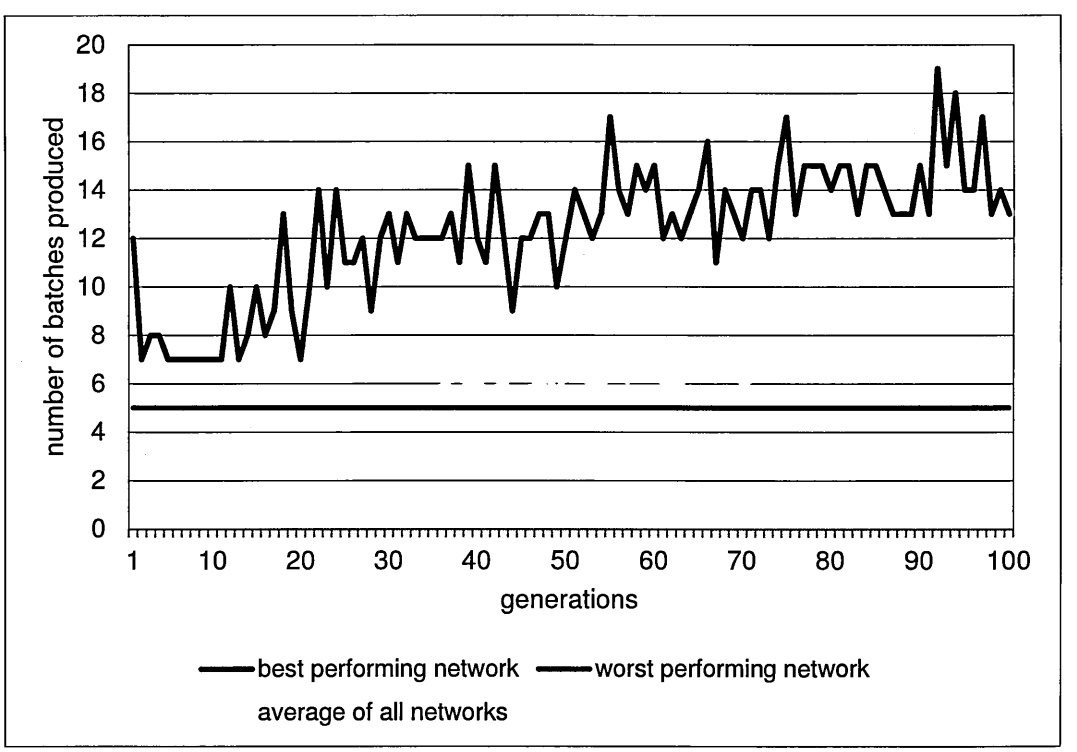


Figure 22: Results for phase 1 run 3

5.3.2 phase 2

(100 generations, 50 networks, Mutation rate 5%)

Runs 1 & 2: the population comprising 10 taken randomly from each of the separate phase 1 populations.

Run 3: the population comprising 10 taken manually from each of the separate phase 1 populations – choosing the best performing individuals.

phase 2, 50 networks, 100 generations, 5400s simulated run time							
run	**mutation rate**	**initial**			**final**		
		worst	**average**	**best**	**worst**	**average**	**best**
p2 run1	5%	5	7.14	14	5	9.44	20
p2 run2	5%	5	9.64	28	5	10.24	19
p2 run3	5%	5	7.26	16	5	9.46	20

Table 11: Results, for phase 2

Table 11 above shows the starting and ending points for phase 2 experimental runs. As for phase 1, the performances during evolution are shown graphically in figures 23-25 below. As before, the blue trace shows the best performing ANN, the yellow trace the averages across the population, and the pink trace the worst performing ANNs.

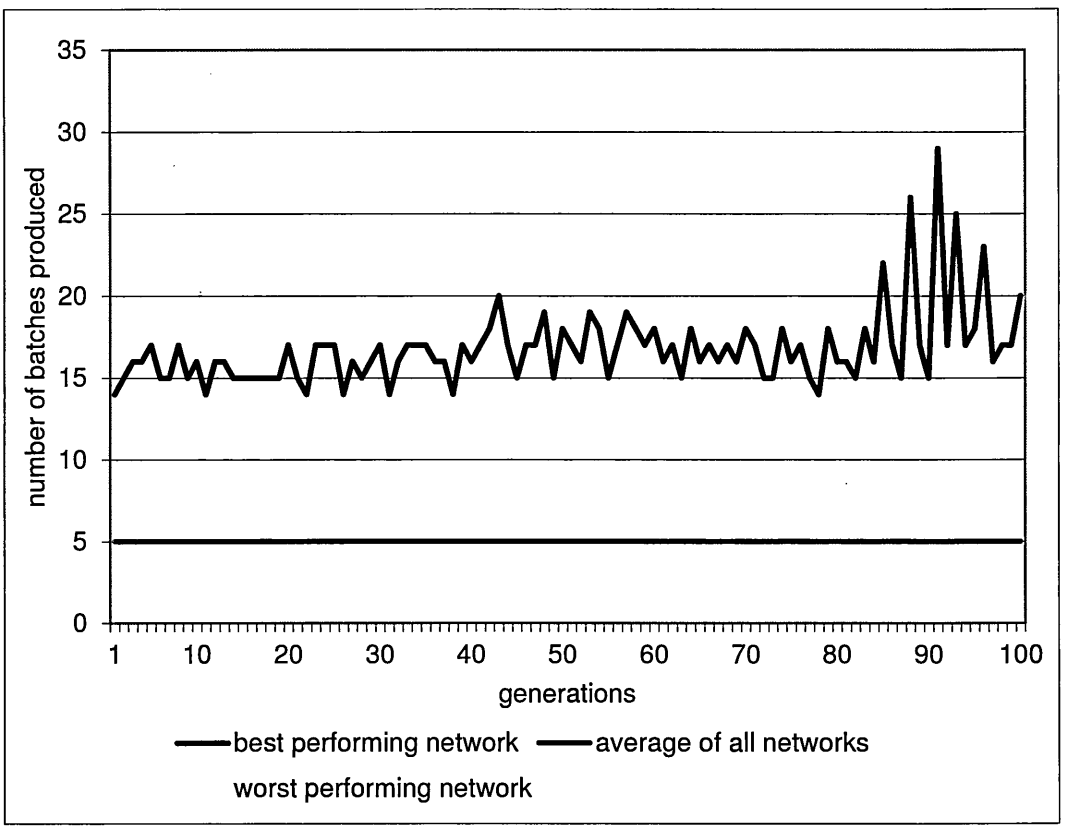


Figure 23: Results, for phase 2 run 1

Figure 23 shows a very gradual rise in the average performance, with the best performing ANN becoming volatile around generation 85. This indicates the EA is abruptly jumping into a new area in the search space perhaps as a result of a mutation of a key parameter. Figure 23 shows something of the nature of an EA as it jumps unpredictably around the search space.

Figure 24 shows a similar process at the start, but after a period of volatility the best ANN settles down. Figure 25 then shows a gradual rise in performance throughout the period.

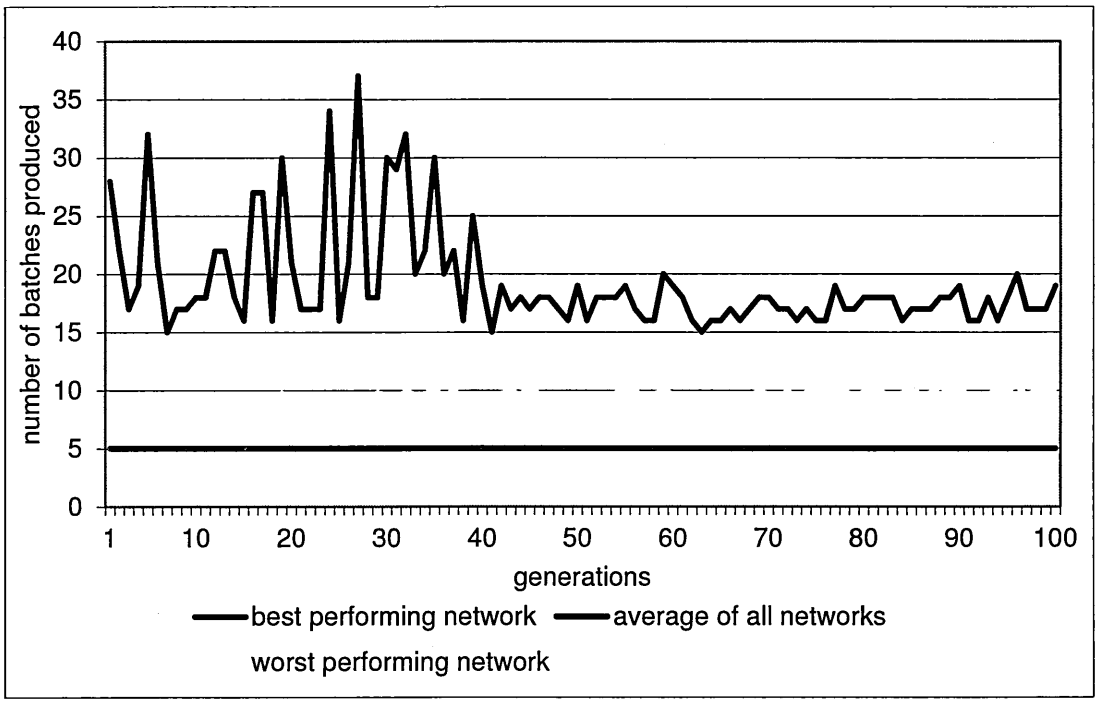


Figure 24: Results, for phase 2 run 2

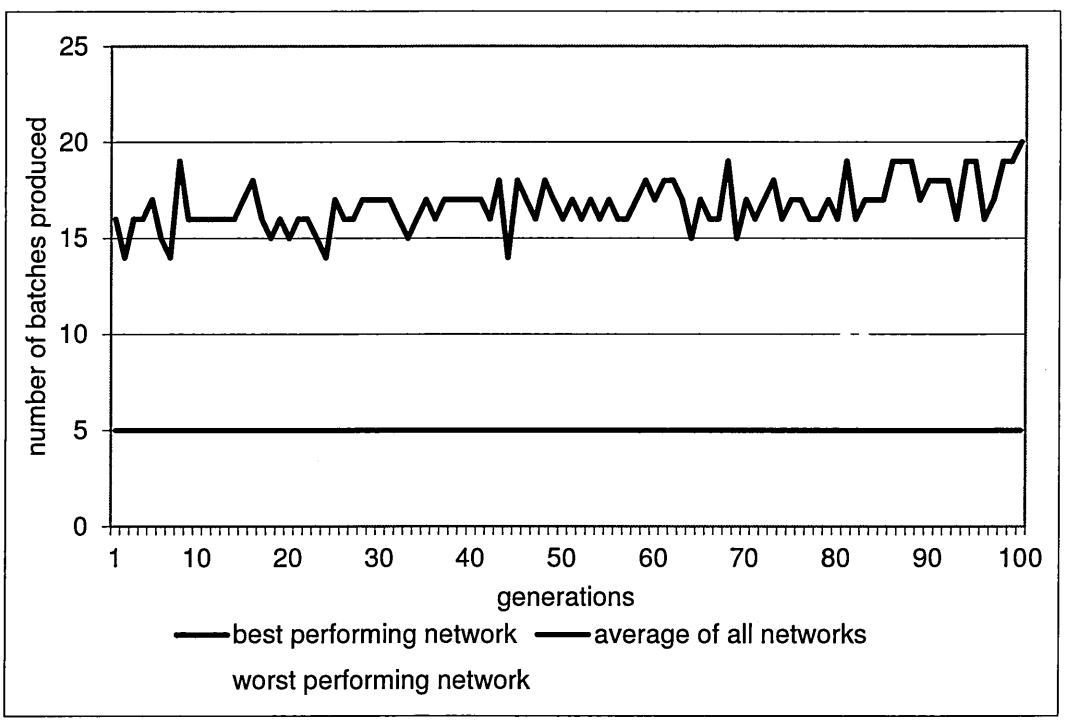


Figure 25: Results, for phase 2 run 3

5.3.3 phase 3

(1 population, 50 individuals taken randomly half each from the populations from the end of phase 2 runs 1 and 3. Mutation 1%. 300 generations)

phase 3, 50 networks, 100 generations, 5400s simulated run time							
run	**mutation rate**	**initial**			**final**		
		worst	**average**	**best**	**worst**	**average**	**best**
p3 run1	1%	5	9.26	17	5	11.36	21
p3 run2	1%	5	10.92	20	5	12.76	43
p3 run3	1%	5	12.48	50	5	12.82	50

Table 12: Results, for phase 3

Figure 26 shows a very gradual rise in performance for the ANNs produced by generations 1-100. This indicates incremental improvement by the evolutionary process.

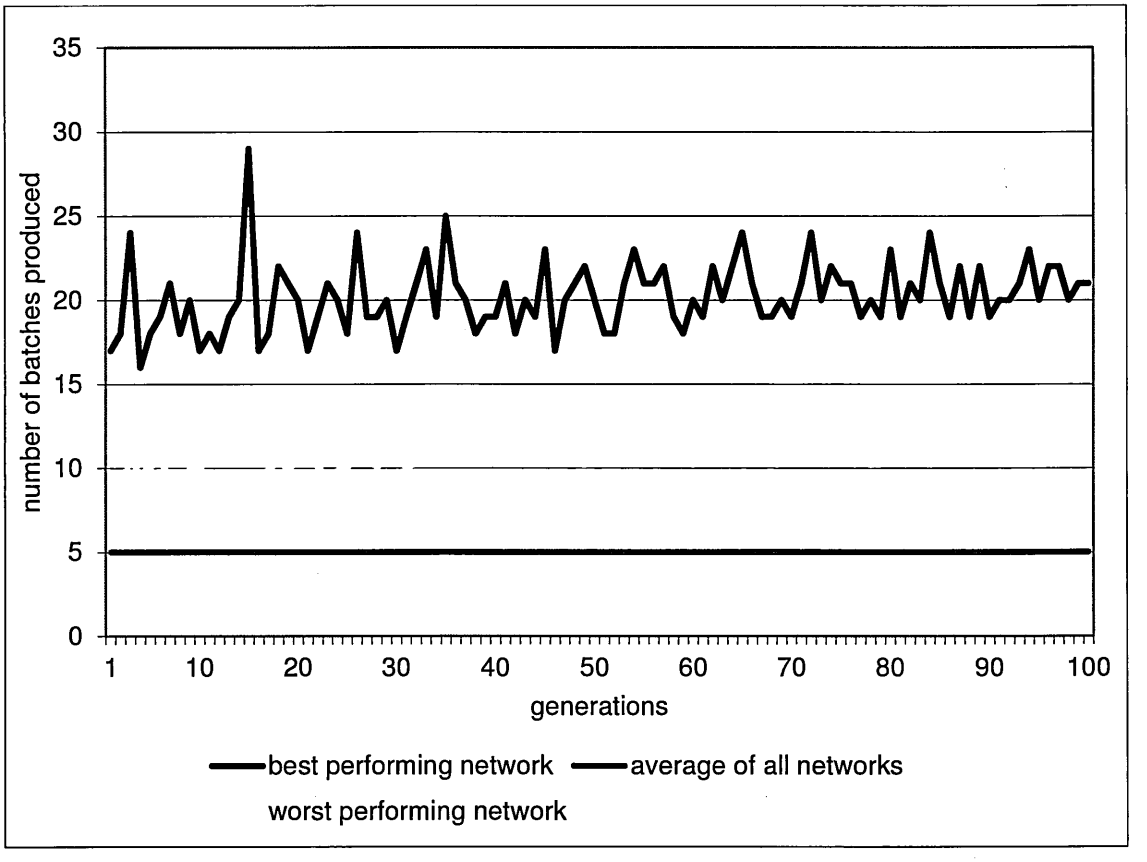


Figure 26: Results, for phase 3 generations 1-100

Figure 27 by contrast shows that at approximately generation 150, ANNs emerge in a different part of the search space and there is a step change in the best performing ANN. Note however that the average performance does not change..

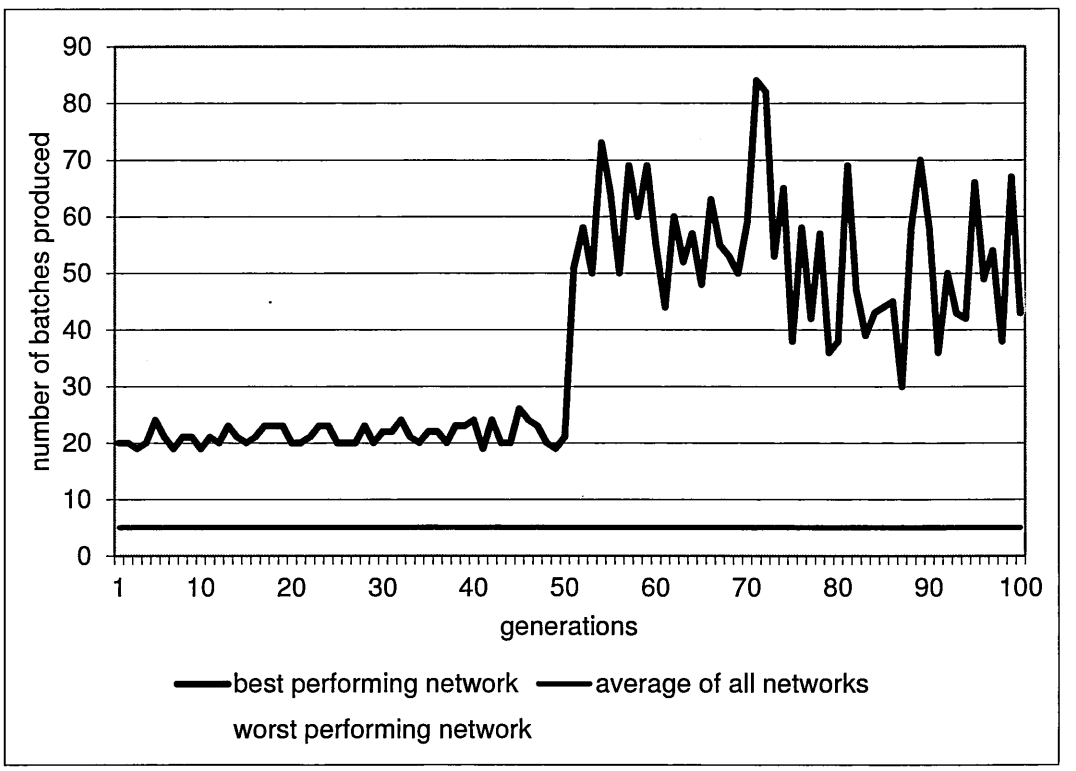


Figure 27: Results, for phase 3, generations 101-200

Figure 27 shows a clear step change in performance. The EA has jumped to a new and better part of the search space. When a mutation or recombination leads to a single network which is markedly better than the rest of the population, then distribution of the features that lead to that performance throughout the rest of the population is very rapid. This offers an insight into the mechanism of the evolutionary process. Figure 28 shows evolution continuing, showing a volatility

in the best performing network but around an average value similar to the higher level (after the step) in the previous figure.

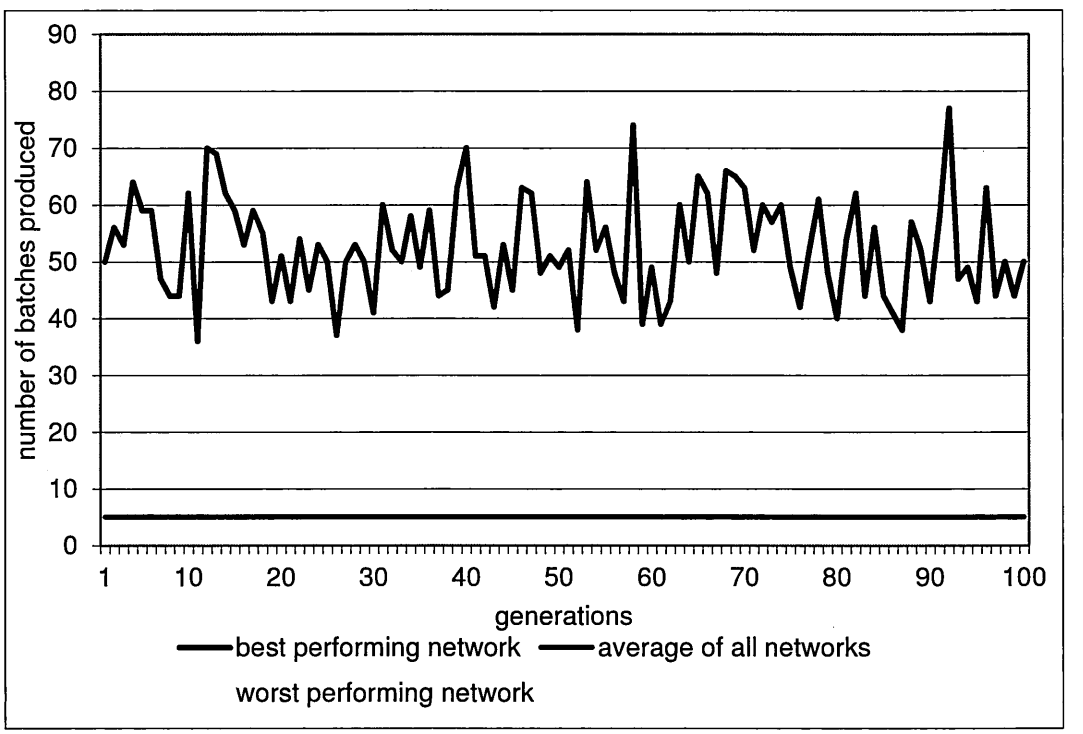


Figure 28: Results, for phase 3 generations 201-300

5.3.4 phase 4

(1 population, 50 individuals from the end of phase 3. 500 generations. Mutation declining from 0.5%)

phase 4, 50 networks, 100 generations for each run, 5400s simulated run time							
run	**mutation rate**	**initial**			**final**		
		worst	**average**	**best**	**worst**	**average**	**best**
p4 r1	0.5%	5	10.88	48	5	14.76	63
p4 r2	0.4%	5	17.90	64	5	29.80	74
p4 r3	0.3%	5	29.40	73	5	35.50	81
p4 r4	0.2%	5	31.90	80	5	35.60	86
p4 r5	0.1%	5	36.80	78	5	36.10	82

Table 13: Results, for phase 4

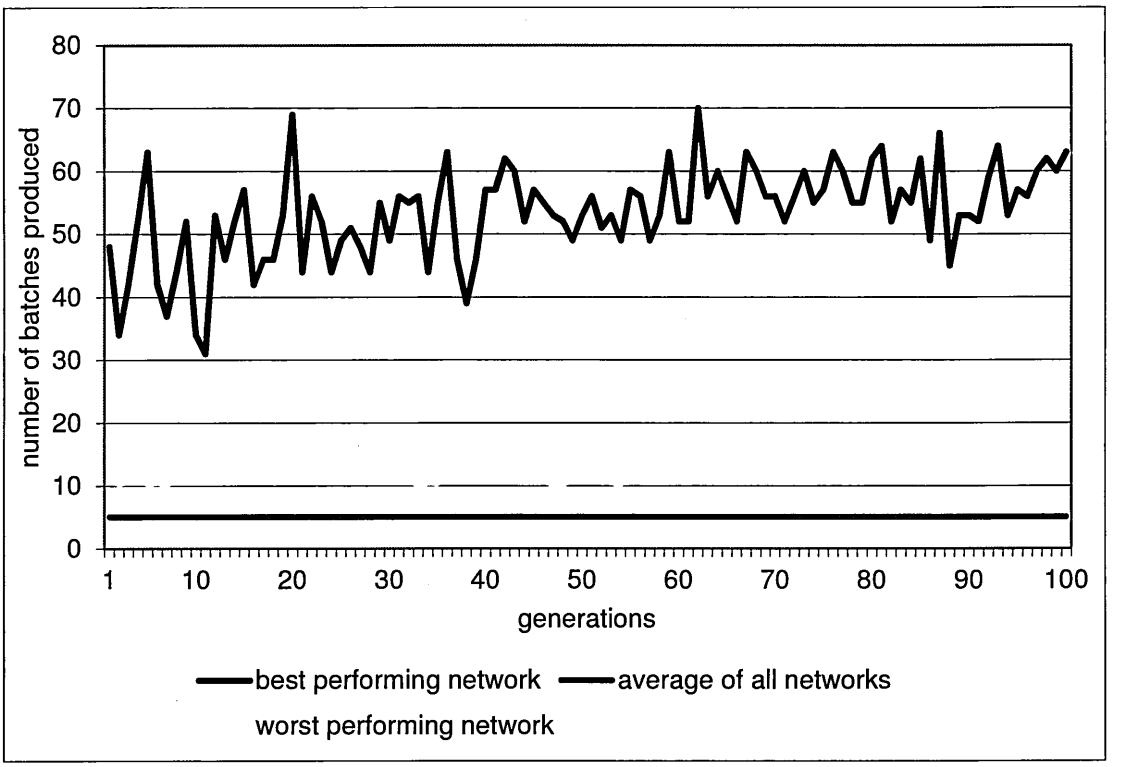


Figure 29: Results, for phase 4 generations 1-100

Figure 29 shows the best performing ANN starting at the level of the end of the previous experiment, and gradually increasing. This is indicative of the results from the generations 101-500. The average also starts to increase as the better performing ANNs replicate across the population. The results from all 500 phase 4 generations are shown in Figure 30. This clearly shows across the longer evolutionary term, a step change in performance around generation 105, and a consequent improvement in performance of the average of the population. It is however noteworthy that the average performance of all networks takes time to increase, this indicating time taken for more effective networks to spread throughout the population.

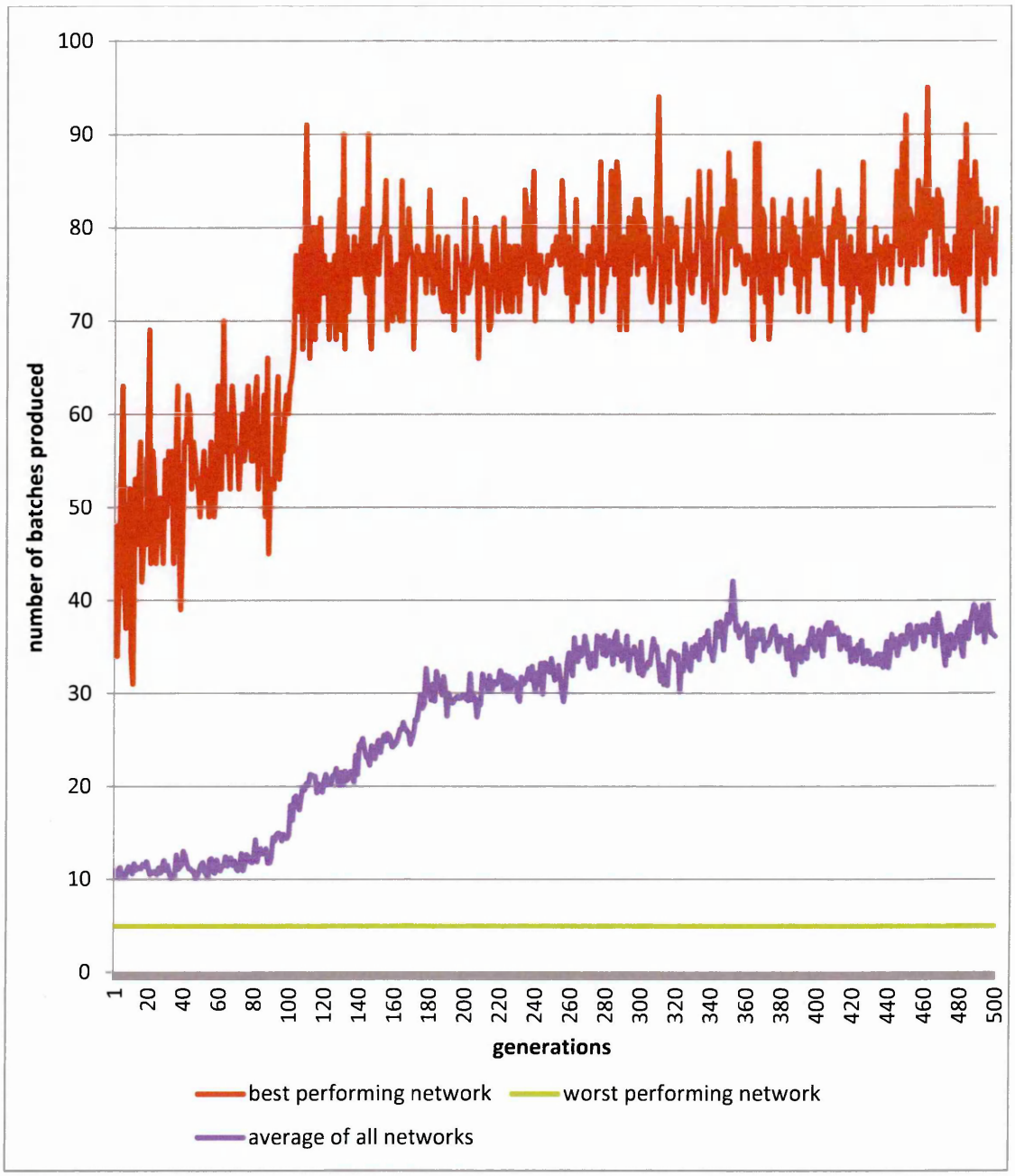


Figure 30: Results from all 500 phase 4 generations

5.4 Conclusion

Figure 31 shows the increase in performance over all the series, with a measure taken at the end of each set of 100 generations. The two curves show the performance of the best individual network in the final generation of the run, and the average of all networks over the final generation.

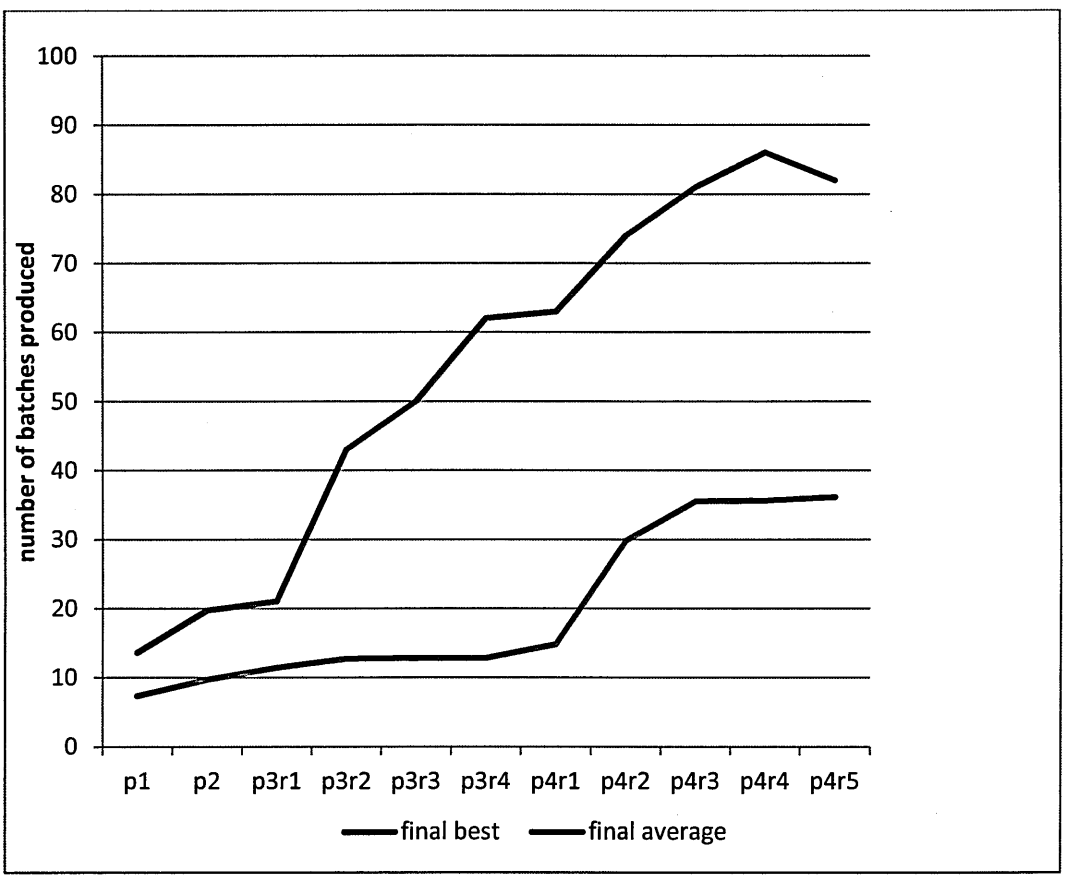


Figure 31: Results summary – all experiments

After the various phases of evolution, the best performing networks are able to reach performances of well over 80, which is very respectable system utilization for the simulated time (1.5 hours). Maximum production would be 90, and conventional control typically achieves 85-90% of this, or 76-81 batches.

Here the evolved controls achieve a best performance of 85 batches, on occasions (due to the mix of work coming through the system). However, conventional control can on occasions achieve the maximum of 90 batches. This is a stochastic system. However the simulation results do indicate the credibility of evolution for generating ANN system controls, which can be comparable to conventional methods.

The experiments do show that the EANN approach could be successfully used in this application. However the experiments also show that there is a need to better understand the implementation of the EANN approach, as the many parameters can mean that it is very difficult to choose the best parameters for a particular application. Therefore it was decided to continue the work by applying EANNs to test problems rather than real world problems.

Chapter Six: EANNs for problem solving I

The following two chapters describe more sophisticated attempts to implement an EANN. Following work in evolving control systems to control industrial equipment, it was realised that two games were similar to this real world problem - a pattern of inputs requires a single output. In order to better understand the effectiveness and method of implementing the EANN approach to real world problems it would be useful to carry out experiments into the EANNs applied to test problems such as the games of Connect-Four and Noughts & Crosses. Here there is the advantage that in certain cases (where a win on the next move is possible) it is very clear that there is an optimum move to be made, so it is relatively straightforward to see if the ANN is able to select this move when presented with this pattern of inputs.

6.1 Introduction

For the implementation described in this chapter, the game of Connect-Four was used. An ANN is used as a black box to make decisions about the next move to make, and an EA is used to find the link weights of a population of ANNs.

Previous research into this type of evolutionary system tends to have the individuals in the population play each other on a random or knockout basis. That approach was followed here.

6.1.1 Connect-4

Connect-Four is a traditional game of strategy. The game is played on a vertical board 7 columns x 6 rows (Figure 32 below) and each player in turn chooses a

column in which to place a counter (each player using either red or blue counters). The counter drops to the lowest available position, each column acting as a stack. If all rows in the column are full then the column cannot be chosen, and if the entire board is full (when 42 moves have been made) and there is no winner, then the game is drawn. The intent for each player is to get four of their counters in a straight line - vertical, horizontal, or diagonal. See Figure 32 to Figure 35.

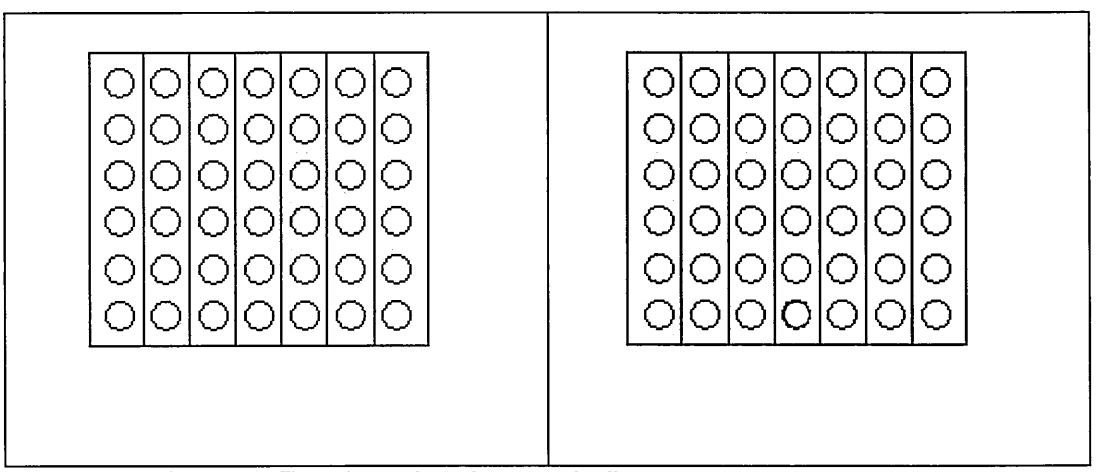


Figure 32: Connect-Four board and example first move

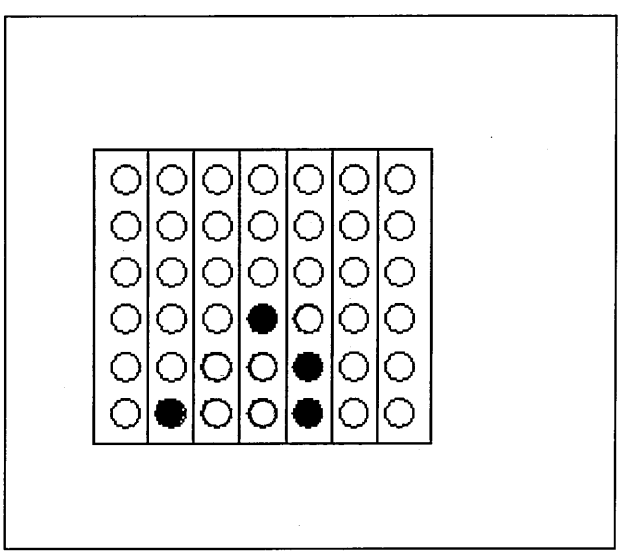


Figure 33: The game after 9 moves

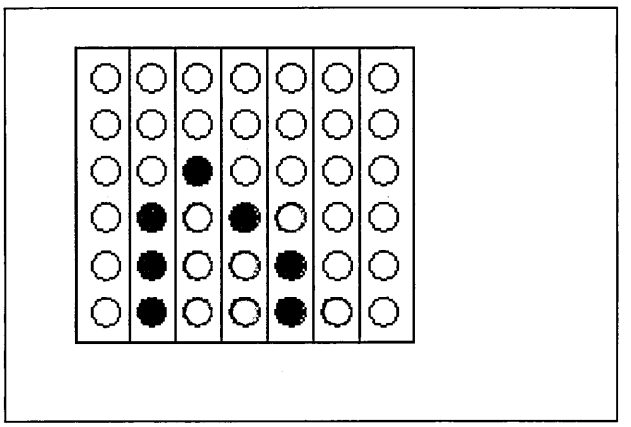


Figure 34: The game after 14 moves.

At the point shown in figure 34, blue cannot fail to win - red must go in column 2 to block blue getting a vertical line of four, but that allows blue to go in the space above, getting a diagonal line of 4

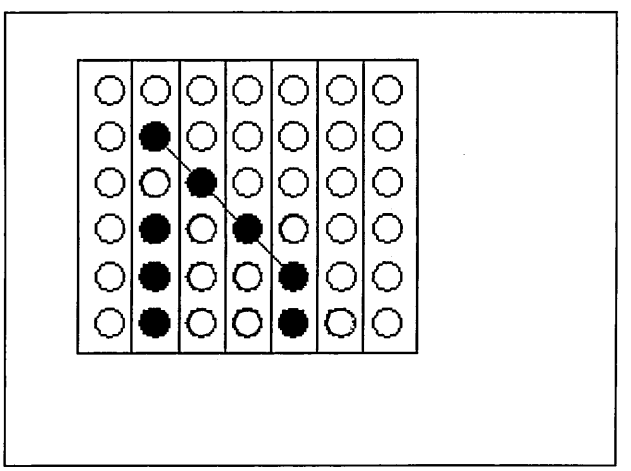


Figure 35: The game after 2 further moves. Blue has won with a diagonal line.

One important complication with this game is that the entire game can translate to the left or right. For example the states shown in Figure 36 below. These games would present completely different patterns to a control system, but represent a virtually identical problem to solve. (The fact that the edges of the area are close - the board is not infinite - mean that the game is different, but in

this case not significantly so.) Any successful control system is therefore required to show some translation invariance.

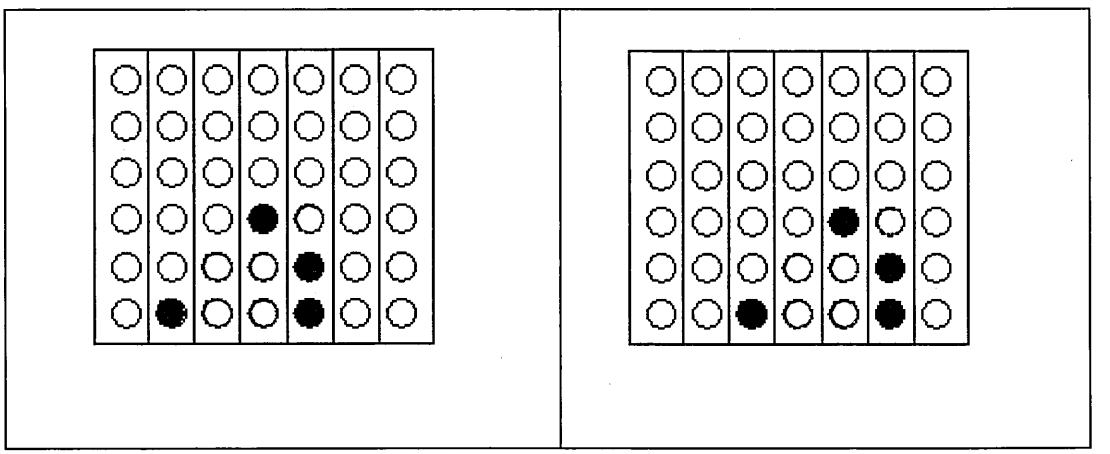


Figure 36: Translation of identical game state

6.2 Method

The game was implemented on a VisualBASIC simulator and a population of 50 individuals generated to the parameters shown in Table 14 below. The network structure and the parameters chosen, were based on experience gained from the work described in chapter 5.

(Generally implementation and methodology was similar to that followed and described in chapter 5)

population	50 individuals
ANN structure	42 inputs 80 hidden units 1 output fully interconnected feed-forward
generations	400
mutation rates	0.5% dropping by 0.1% each 100 generations
performance assessment	each network taken in turn to play 20 games, each against a randomly selected opponent. A score awarded for each game as (-1) for a loss, (0) for a draw, and (+1) for a win. Therefore each network is given a performance score as -20 << +20
EA strategy	calculate average performance and find best performer best performer always retained for each ANN, if performance < average, then 50% chance of replacement replacement is by single point crossover, with random crossover point, using two randomly selected ANN parents from the pool of ANNs having better than average performance

Table 14: Parameters for generation of population of ANN individuals

At the end of each generation, the performance of each ANN was measured. 20 games were played against a randomly selected opponent network. A simple payoff function of {1, -1, 0} (for win, lose, draw) was used. (Chapter 7 describes a more sophisticated look at the payoff function.)

Each network therefore was awarded a score of -20 to +20, and the average score of all the networks was recorded at the end of each generation.

6.2.1 Diversity

Successful networks generate offspring networks. It was expected that certain sequences of numbers in the genetic sequence will become reproduced across the population (as predicted by the Schema Theorem, (Holland, 1975)). The effect of this is to reduce diversity in the population. It was decided therefore to attempt to measure the diversity of the population.

A measure of genetic diversity of the population was implemented as follows:

- each ANN comprises a sequence of 3440 numbers.
- each ANN is compared to each other ANN and a count made of the occasions where the sequence numbers match.
- diversity measure is a % probability of each sequence number being unique in the population.

6.3 Results

Figure 37 shows the variation of average performance for each generation.

Figure 38 shows the percentage diversity for each generation according to the method of measurement described above.

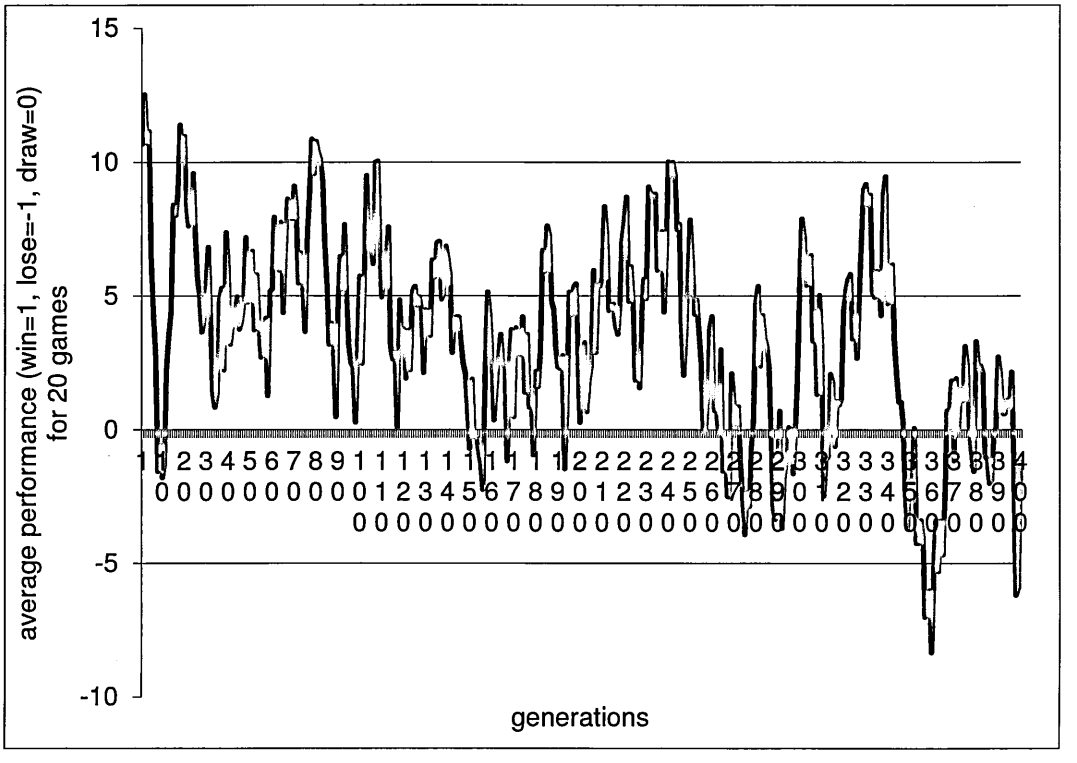


Figure 37: Variation of average performance for each generation

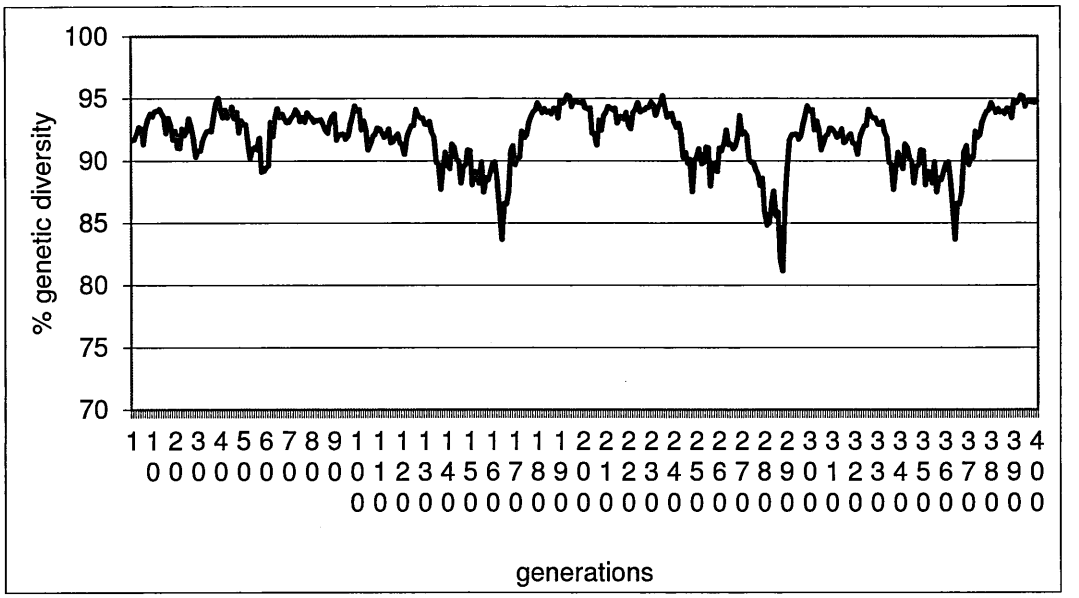


Figure 38: Genetic diversity (%) by generations

6.4 Conclusions

On examination of the results, it was realized that the choice of measurement in this experiment was poor. Because each network was tested against other networks at the same stage in evolution, the *average* performance would not in fact be expected to show any overall improvement. To overcome this, at various stages through the experiments, the best performing network was taken out and tested by playing manually against a human player: no behavior other than moving apparently at random, or continuously in the same place was observed. This lack of success, could be attributed to one or more of the following:

- the evolutionary period (number of generations) was not long enough to find ANNs able to effectively achieve the goal, and therefore running the evolutionary process for longer would overcome this.
- the scale of the ANNs (number of units) was too small, and the ANN simply does not have the necessary capacity to span this problem.
- there was an error with the strategy or the implementation.

Genetic diversity was a useful measure. If for example the evolutionary process generated a particularly successful individual then the number sequences in that individual's chromosome would be replicated across the population and the diversity measure would be expected to decrease. It *is* observable from Figure 37 and Figure 388 that at various points (around generations 160, 290 and 370) both the average performance *and* the genetic diversity declines, and this would be expected: while the Schema of a slightly more successful individual becomes widespread in the population, the diversity will decrease, and so too will the average performance as ANNs tend to be played against other ANNs more and

more similar to themselves. The average performance will drop to zero. However, if mutation results in a slightly more successful ANN, then the average performance will rise (in the same way that in a group of 20 people all 1.8m tall, with one exception 2.0m tall, then the average is greater than 1.8m).

6.5 Discussion

Recalling Figure 27 from chapter 5, which illustrates the potential for an EA to suddenly find a successful part of the search space, it cannot be ruled out that had evolution been allowed to continue for a further generation, then a successful individual may have been generated. However this is a stochastic process. It is equally possible that evolution may continue for thousands of generations with no improvement. This experiment illustrated a weakness in this EANN approach: given that performance does not appear to be improving, it is not possible to say with confidence that it is the EA or the ANN that is the limitation, and it is equally not possible to determine exactly when to stop evolution on the grounds that performance will *never* improve.

This experiment showed that a more sophisticated use of the payoff function is required. That is, one which weights the relative importance of winning, as opposed to not losing. It was decided that it would also be useful to look at precursor states to a win (for example, as shown in Figure 34). If the network can recognize a state where a win is possible with the next move, then the next move should always be to take the win. This led to the experiment which is described in chapter seven, where paired inputs were used, and also a static benchmark of performance in order that the average performance can be seen to rise (or not).

An important conclusion from this set of experiments with relation to the application to real world problems (such as the LSC conveyor system from chapter five) is that a poor choice of parameter such as payoff function, or method of measuring success will probably lead to the system implementation not being successful. However, this may not be apparent without the most rigorous of analysis and as a result significant amounts of time and effort may be wasted.

Chapter Seven: EANNs for problem solving II

This chapter describes a development of the attempt to implement an EANN following on from chapter 6. Another game was used - noughts & crosses – where, similar to connect-four, a pattern of inputs requires a single output. In this case – following the conclusions from chapter 6 – a more sophisticated payoff function was used, and also a deterministic player was introduced which meant that there was a better means of measuring the performance of the population of networks and therefore measuring the rise in performance due to the evolutionary process.

Part of this work has been previously published in (Morley, 2009)

7.1 Introduction

This chapter explores the use of a hybrid system in the context of playing games where a pattern of inputs (current state) is presented to the system, and the output required of the system is a decision on the next state (where to go).

In this experiment, the game - noughts & crosses (tic-tac-toe) - was used. An ANN is used as a black box to make decisions about the next move to make, and an EA is used to find the link weights of a population of ANNs.

In particular this experimentation looked at the effect of different payoff functions (for win, draw, and lose) using an approach similar to that taken in (Fogel, 1993).

Previous research into this type of evolutionary system tends to have the individuals in the population play each other on a random or knockout basis. That was the case here except that there was also a chance (a variable probability) that a deterministic (i.e. not an ANN) algorithm would be used. This was introduced in order that the average performance of the population could be seen to improve. (See discussion in section 6.4).

A further refinement of the approach from chapter 6, was that inputs to the network were *pairs* of the game spaces representing precursor states to a win.

7.2 Method

Noughts & Crosses (Tic-tac-toe) is a traditional game of simple strategy. Two players attempt to get a line of three counters taking alternate turns on a 3x3 board.

The game was implemented on a VisualBASIC simulator, and a population of ANNs generated as shown in Table 15 below, using parameters from experience gained in the experiments described previously.

1. Static Parameters	
ANN	structure - fully interconnected feed-forward 4 layer, similar to that shown previously in Figure 21. 48 inputs 80 hidden units layer 1

	50 hidden units layer 2 9 outputs (8290 link-weights in total) transfer - weighted sum including bias, with sigmoid function
EA	population size 50 generations 2000 mutation rate 1% x (0.9)^(no. of generations) recombination by single point crossover, random split point parents - selected randomly from pool of >average performance
performance assessment	each network taken in turn to play 10 games, each against a randomly selected opponent from the ANN population, or the possibility to play a deterministic automatic player instead of another ANN Payoff function = {variable} (score awarded for each loss, draw, or win.) For each generation each ANN is given a score comprising the sum of payoffs for each of the 10 games. This score is used in the EA strategy.
EA strategy	calculate average performance and find best performer

	- best performer is always retained for each ANN, if performance < average, then variable chance of replacement replacement is by single point crossover, with random crossover point, using two randomly selected ANN parents from the pool of ANNs having better than average performance
2. Variables	
ANN	link weights - initialized with random weights
EA	payoff function {1,-10,0}, {10,-1,0} chance of replacement if score < average, 50%, 95% chance of playing deterministic player, 50%, 95%

Table 15: Summary of experimental method

The 48 inputs comprised pairs of game-spaces, examples of which are shown in Figure 39 below.

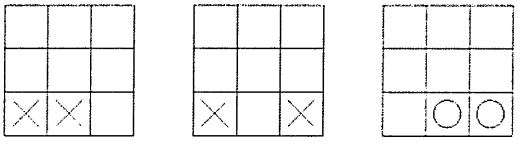


Figure 39: Typical inputs to the network, comprising states of pairs of spaces.

Figure 39 shows three of the 48 permutations of states which precede a win (or lose). Each of the 48 possible win-precursor states was given as an input to the network. The first two illustrated were assigned a value of 1 (precursor to a win), and the third a value of -1 (precursor to a loss). It was necessary that the third space was empty, but the remaining spaces were 'don't cares'. Previous experiments (not reported here) had carried out similar work without the use of win-precursor inputs. Results were in fact similar to those reported here.

7.2.1 Deterministic player

There was also the chance of playing the deterministic player which operated as follows;

- if a win can be obtained then go there,
- otherwise go in a random location.

The actual chance of the deterministic player being used was variable. It was thought useful to find out if this would be beneficial or not, to the evolution of a workable solution.

Note that the deterministic player intentionally implemented a substantially sub-optimal solution.

7.3 Results

8 sets of experiments were conducted. At the end of each run, the networks in the population were tested against a specific deterministic algorithm which operated according to the following rules:

- ANN moves first, first replying move taken in each of the remaining places in turn, then;
    - if a win can be obtained then go there,
    - if a win for the opponent can be blocked then go there,
    - otherwise go in a random location.

Thus 9 results were obtained for each network (one result for each match against another ANN, and one for the match against the deterministic player). Table 16 below records the number of those games which were won or lost by the network, or drawn. Furthermore for comparison a randomized set of ANNs were also tested.

run	payoff function	probability: replacement if <average	probability: playing auto-player	win	lose	draw	%win	%lose
1	{10,-1,0}	0.95	0.95	40	298	112	8.89%	66.22%
2	{1,-10,0}	0.95	0.95	16	299	135	3.56%	66.44%
3	{10,-1,0}	0.50	0.95	36	328	86	8.00%	72.89%
4	{1,-10,0}	0.50	0.95	29	286	135	6.44%	63.56%
5	{10,-1,0}	0.95	0.50	30	323	97	6.67%	71.78%
6	{1,-10,0}	0.95	0.50	32	292	126	7.11%	64.89%
7	{10,-1,0}	0.50	0.50	58	270	122	12.89%	60.00%
8	{1,-10,0}	0.50	0.50	34	268	148	7.56%	59.56%
random	n/a	n/a	n/a	24	311	115	5.33%	69.11%

Table 16: Results for each network

The following table re-orders the results with most successful (highest win %) first:

run	payoff function	probability: replacement if <average	probability: playing auto-player	win	lose	draw	%win	%lose
7	{10,-1,0}	0.50	0.50	58	270	122	12.89%	60.00%
1	{10,-1,0}	0.95	0.95	40	298	112	8.89%	66.22%
3	{10,-1,0}	0.50	0.95	36	328	86	8.00%	72.89%
8	{1,-10,0}	0.50	0.50	34	268	148	7.56%	59.56%
6	{1,-10,0}	0.95	0.50	32	292	126	7.11%	64.89%
5	{10,-1,0}	0.95	0.50	30	323	97	6.67%	71.78%
4	{1,-10,0}	0.50	0.95	29	286	135	6.44%	63.56%
random	n/a	n/a	n/a	24	311	115	5.33%	69.11%
2	{1,-10,0}	0.95	0.95	16	299	135	3.56%	66.44%

Table 17: Results reordered by success rate

7.4 Conclusions

It was proposed that evolution as a general approach will tend to solve problems such as this, where no information regarding the object of the game or training as to appropriate moves was given. Given the present state of the board, the ANN had to decide on the best next state, and no feedback was given until the end of the game, at which point feedback was limited to 'win / draw / lose' as outlined.

It was further considered that while the evolutionary process is robust, there are parameters which will affect the speed of finding an effective solution. Three parameters were varied in the course of this experiment. There are many more parameters, and it would be useful further work to establish their relative effect.

7.4.1 Payoff function

The payoff function for win / draw / lose is important, and in particular that it is asymmetric. The actual figures are irrelevant, but the *relative* values are.

Two payoff functions were tested with the following expectation;

- {10, -1, 0} strong advantage for winning - expect higher proportion to win
- {1, -10, 0} strong disadvantage for losing - expect higher proportion to win or draw

From the ordered table it can be seen that the first payoff resulted in the more successful individuals as expected.

7.4.2 Probability of replacement if below average performance

This represents the harshness of the environment. The probability that an individual will be eliminated from the population if its performance is below average.

Two probabilities were tested with the following expectation;

0.95 : strong chance of elimination

0.50 : weaker chance of elimination

From the ordered table it can be seen that the weaker chance of elimination resulted in the more successful individuals – not as expected – but perhaps because a harsh environment eliminates individuals too soon - before they have chance to develop successfully.

7.4.3 Probability of playing the deterministic player

The deterministic player will always take direct opportunity to win. Therefore we would expect that the ANNs with more exposure to this player, to be more effective at blocking wins.

Two probabilities were tested with the following expectation;

- 0.95 : strong chance of facing deterministic
- 0.50 : weaker chance of facing deterministic

The results on this did not give a consistent indication that this parameter was important.

7.5 Discussion

The best approach to problem optimization is problem dependent - according to the no free lunch theorem (Wolpert & Macready, 1997). This also holds for the initial conditions and parameters. The best parameters for one problem will not necessarily be the best for another.

Over the experiments described in chapters 3 to 7, it was found that the EANN approach could be successfully used in the decision making part of the control of linked sequential systems. That said, the more abstract experiments (chapters 6 and 7) showed that the exact method of implementation and initial parameters needed to be chosen with care. Therefore it may not be possible to find an EANN approach that is effective in *all* situations for practical implementation (this is in keeping with the No-Free-Lunch theorem, (Wolpert & Macready, 1997).

The implication for industrial laundry systems, and linked sequential systems in general, is that there is scope for the application of EANN based controls, and these can offer significant benefits. However, the implementation needs to be planned with care taking into account the lessons learnt in these sets of experiments, and a single EANN system will not be applicable in all situations but rather will have to be planned individually according to the circumstances. The experiments have shown that it is not futile attempting to implement an EANN system, but neither is it necessarily simple.

Chapter Eight: An Agent Based Model for dynamic modelling of a linked sequential system

Following the work described in the previous chapters, it was observed that there was a lack of rigorous understanding of the performance of these industrial wash systems (which are simply sequences of machines feeding from one to another).

In any analysis of the performance of a system, it is essential that there is a rigorous method of modelling the system that is good enough for the analysis required. In the industrial laundry field this modelling is lacking, and therefore it was considered that it would be valuable research to find better methods of modelling the performance of these systems.

8.1 Introduction

In a linked sequential system (such as shown in Figure 2), the slowest machine is generally seen as the rate determining step, setting the rate of the whole system. In fact the slowest machine is actually the maximum-rate determining step and sets the upper limit on the rate of the system.

In a linked sequential system in an industrial laundry, the most expensive part of the system is the batch tunnel washer (see section 3.3) and the operator will require it to be utilized as fully as possible. Therefore multiple dryers are usually required, for example as shown in Figure 400 (Figure 4 repeated)

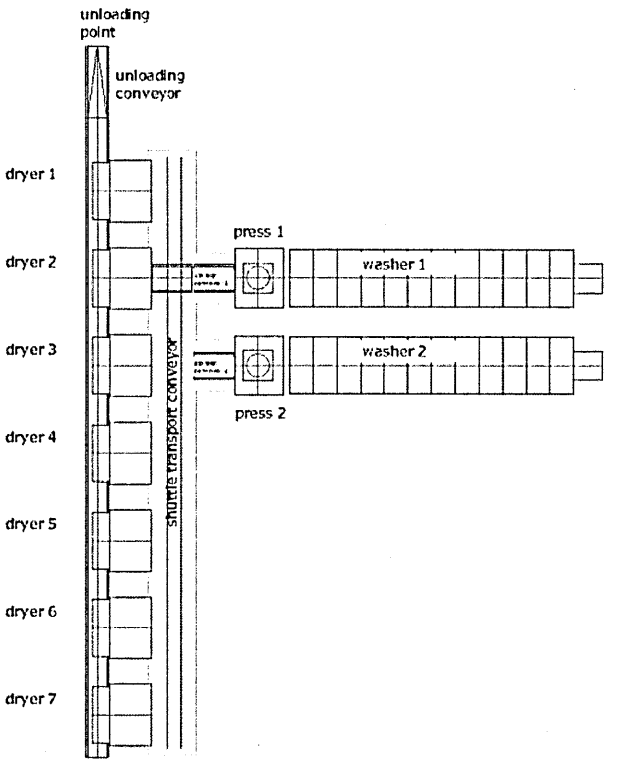


Figure 40: Typical layout of washing system (Figure 4 repeated)

The only part of Figure 40 where a slightly more complex control decision has to be made is the implied element between the press and the dryers, which has to decide which dryer to take the batch of work to, (assuming more than one is empty).

This suggests that an agent based approach is suitable, where each element can be modeled as a fairly simple agent, and the performance of the system as a whole emerges from the interaction between these agents.

This suitability was further suggested by (Parunak, et al., 1998) which proposed that ABM is a suitable approach in domains which are characterized by a high

degree of localization and distribution and dominated by discrete decisions, which is the case here.

(Parunak, et al., 1998) also suggests that the alternative-paradigm – Equation Based Modelling - remains popular because of the ready availability of tools at the easy disposal of the practitioner.

8.2 Method

The system shown in Figure 40 is a typical industrial washline. There are two washers, each with a press and a batch storage conveyor. There is then a single shuttle conveyor which takes batches from either of the washers and loads them into one of the available dryers.

A model of this system was implemented in software as VisualBASIC code embedded in a spreadsheet (MS Excel) running on a normal office computer. A screenshot of the main page is shown as Figure 41. The spreadsheet format allowed for easy entry and adjustment of the many variables, and user-friendly interface. The VisualBASIC code can read the variables directly from the spreadsheet. The main routine is then listed in Table 18 as a sample of the code. The various sub-routines are not quoted here.

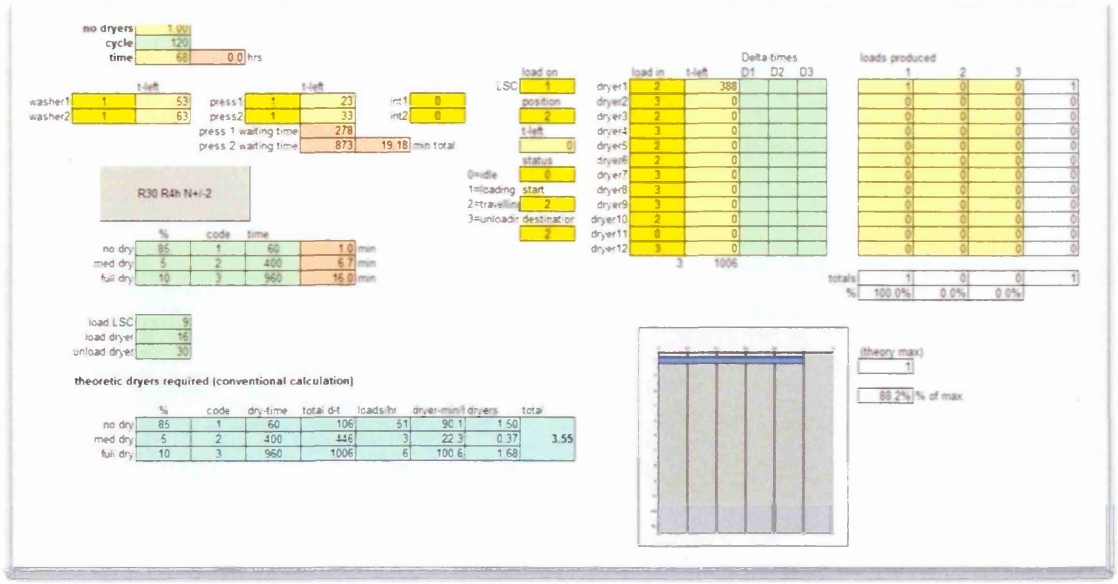


Figure 41: Screenshot of simulator

```
For zzz = 10 To 45 Step 5

For yyy = 5 To 55 Step 5

xxx = 100 - (yyy + zzz)



Excel.Worksheets(1).Cells(17, 3) = xxx

Excel.Worksheets(1).Cells(18, 3) = yyy

Excel.Worksheets(1).Cells(19, 3) = zzz



'dryers

dry_n = Int(Excel.Worksheets(1).Cells(43, 10))

For dryers = dry_n - 2 To dry_n + 2

Excel.Worksheets(1).Cells(2, 3) = dryers

   Call reset

   'run 30 min
```

```
    For f = 1 To 1800
    Call Run_1sec
    Next f
    'reset loads produced
    Call reset_loads
    'run 4 hour
    For f = 1 To 14400
    Call Run_1sec
    Next f
    'record results
    Call record
Next dryers
Excel.Worksheets(1).Cells(2, 3) = dryers
Next yyy
Next zzz
End Sub
```

Table 18: Main simulator routine

The behaviour of each unit was as described below.

8.2.1 Operation of Each Unit

- washers
    - 2 instances.
    - each washer would have two parameters, the category of work in the final compartment and the remaining time. On the expiry of the time,

the washer will look forward to the associated press and if the press is empty, it will unload, else it will wait. On unloading, it will start again with the full cycle time.

  - washer cycle time, variable parameter, typically set to 120 sec.

- presses
  - 2 instances, 1 associated with each washer
  - each press again has two parameters; the category of work being pressed, and the remaining time. The cycle time here would be 75% of the washer cycle time, this being typical in real systems, so that generally the press *should* be waiting empty for the washer to unload.
- intermediate conveyors
  - 2 off, 1 associated with each press
  - each conveyor has only a single parameter, the category of work in the batch being stored, or empty. These conveyors decouple the press from the shuttle conveyor and have the simplest control system. Their goal is simply to move work on as fast as possible.
- shuttle conveyor
  - 1 shuttle only, serving both washers and all dryers.
  - with parameters
    - category of work currently held (or empty)
    - current position
    - destination position
    - time remaining in current action
    - status - e.g. idle, loading, travelling, unloading
- dryers
  - a variable number from 1-14 could be handled by the simulation

- parameters are
  - category of work currently held (or empty)
  - time remaining in current action
  - loads produced of the three different types

8.2.2 Timings

The most important input from the point of view of calculating the performance and capacity of the system is the relative proportion of 3 different work categories (by drying programme). For example, Table 19 gives the proportion of the categories, and the associated drying times. The drying time was the only difference between the categories although in reality these would generally be subject to different wash processes too.

	%	code	time /s
no dry	0	1	60
med dry	55	2	400
full dry	45	3	960

Table 19: Proportion of different work categories

Table 20 shows the times for each shuttle loading action.

	time / s
load shuttle conveyor	9
load dryer	16
unload dryer	30

Table 20: Times for loading actions

Table 21 shows times for the shuttle conveyor to travel between positions. In this experiment this was calculated mathematically as 8 seconds per horizontal position travelled plus 6 seconds for a lift or lower operation. However, the variable nature of this lookup table allows for any real system to be modelled following timing of the actual travel times. This includes for the fact that this table is not necessarily symmetrical about the diagonal - i.e. travel time from A to B is not necessarily the same as travel time for B to A, although in this set of travel times, the travel times *were* symmetrical.

timings	start 1	2	3	4	5	6	7	8	9	10	11	12	13	14
	W1	W2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12
dest 1 W1	0	8	14	6	14	22	30	38	46	54	62	70	78	86
2 W2	8	0	22	14	6	14	22	30	38	46	54	62	70	78
3 D1	14	22	0	8	16	24	32	40	48	38	46	54	62	70
4 D2	6	14	8	0	8	16	24	32	40	48	38	46	54	62
5 D3	14	6	16	8	0	8	16	24	32	40	48	38	46	54
6 D4	22	14	24	16	8	0	8	16	24	32	40	48	38	46
7 D5	30	22	32	24	16	8	0	8	16	24	32	40	48	38
8 D6	38	30	40	32	24	16	8	0	8	16	24	32	40	48
9 D7	46	38	48	40	32	24	16	8	0	8	16	24	32	40
10 D8	54	46	56	48	40	32	24	16	8	0	8	16	24	32
11 D9	62	54	64	56	48	40	32	24	16	8	0	8	16	24
12 D10	70	62	72	64	56	48	40	32	24	16	8	0	8	16
13 D11	78	70	80	72	64	56	48	40	32	24	16	8	0	8
14 D12	86	78	88	80	72	64	56	48	40	32	24	16	8	0

Table 21: Shuttle travel times

8.2.3 Other Variables

Table 22 summarises the general system parameters. Table 23 then gives the variables which were fixed or altered in this set of experiments.

parameter	name	type	range / notes
n	number of dryers	integer	1-12
c	washer cycle time	integer	potentially 1+, but to represent a practical system, 90-240
x:y:z	work mix	3 x integer	% work mix such that x + y + z = 100%
tx ty tz	dry time (x/y/z)	3 x integer	dry time (seconds) for each work type (x, y, z), potentially 1+, but to represent a practical system 30-1200
LLSC	load LSC	integer	time to load the LSC, potentially 1+, but to represent a practical system 5-30
LD	load dryer	integer	time to load a dryer, potentially 1+, but to represent a practical system 5-30
UD	unload dryer	integer	time to unload a dryer, potentially 1+, but to represent a practical system 15-60
T(14,14)	travel times	matrix of integers	each element represents the time in seconds for the LSC to travel from Start to Destination points. Each potentially 0+, but to represent a practical system 0-100

Table 22: General system variables

parameter	name	type	range / notes
n_c	number of dryers	variable	where n_c is the conventionally calculated number required, and truncated to an integer - n
n	number of dryers (integer)	variable	where n_c is the conventionally calculated number required n varied as n_c +/- 2
c	washer cycle time	fixed	120 (a typical mid-range value for a practical system)
x:y:z	work mix	variable	the longest two dry times were those for y & z, and the highest possible combination without requiring more than 12 dryers (the software limit) was z=45, y=55, x=0. Therefore the values were varied as; z = 10 to 45 in steps of 5 y = 5 to 55 in steps of 5 x = 100 - y - z
tx ty tz	dry time (x/y/z)	fixed	tx = 60 ty = 400 tz = 1060, typical system values
LLSC	load LSC	fixed	9
LD	load dryer	fixed	16
UD	unload dryer	fixed	30
T(14,14)	travel times	fixed	varying from 0-86, symmetrical (in a practical system the travel times need not be symmetrical)

Table 23: Fixed and variable parameters

For each run, the input was the mix of work; x:y:z. For example, x=30% sheets, y=25% garments, z=45% towels, requiring none, part, and full drying respectively). This mix was used to calculate n_c, the number of dryers required using the conventional calculation. This was rounded down to an integer, and then 5 runs were executed with the number of dryers varying from (n_c - 2) to (n_c + 2). This was because while the conventionally calculated number of dryers n_c is not the ideal number, it is generally not too far out, and the absolute ideal will always be within 2 of n_c.

However in the actual experiment, loads were selected at random, using x:y:z as the probability distribution. Therefore over a certain experimental run there was a difference between:

x:y:z the probability distribution for loads selected

x':y':z' the actual ratio of different loads

After each run, x':y':z' was used to calculate m, the number of dryers using the conventional calculation (which varied slightly from n_c).

For each run, 4 hours of productive time was simulated. The key output was the total number of loads produced (LP), both in real terms and as a percentage of the total number that *could* have been produced if the dryer system was able to take every single load potentially produced by the washers (30 per machine per hour for a 120 second cycle time).

The actual output of the system (LP%) was then corrected by the factor m/n_c

Table 24 below summarises the various outputs that were taken from each run of the simulation.

parameter	name	type	range / notes
LP	number of loads produced	integer	For 2 hours production and c=120, the theoretical max is 120 because two washers with a cycle time of 120 seconds will produce 30 loads per hour each. LP=total number of loads produced by each dryer. Number of loads produced by each dryer was recorded but not considered to be a key output
LP%	number of loads produced %	%	LP / 120 * 100%
x':y':z'	work mix	3 x rounded to 2dp	% work mix such that x' + y' + z' = 100%
m	number of dryers based on x'		if the actual work mix is used as the input to the conventional dryer calculation, m will differ from n.

Table 24: Outputs for each run

8.3 Results

Figure 42 below shows the main output of the series of experiments. Each trace on Figure 42 shows the results from each separate experimental sequence, where the same input data was used, varying the number of dryers from n-2 to n+2.

The general trend is clear. Figure 43 then shows the average of all these experiments.

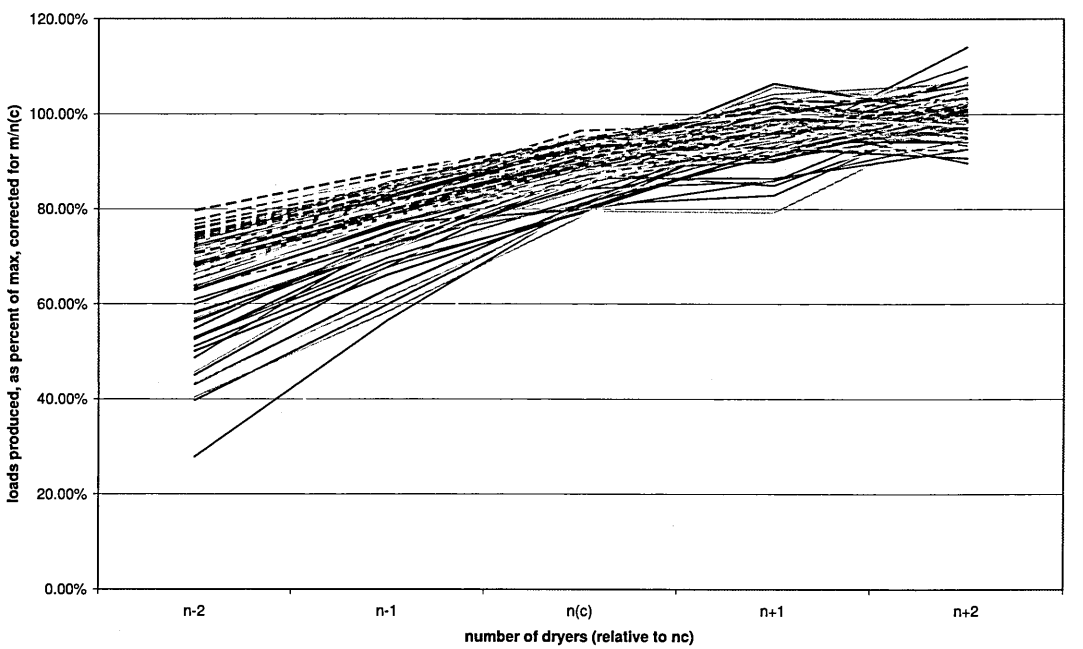


Figure 42: Results of all the experimental runs

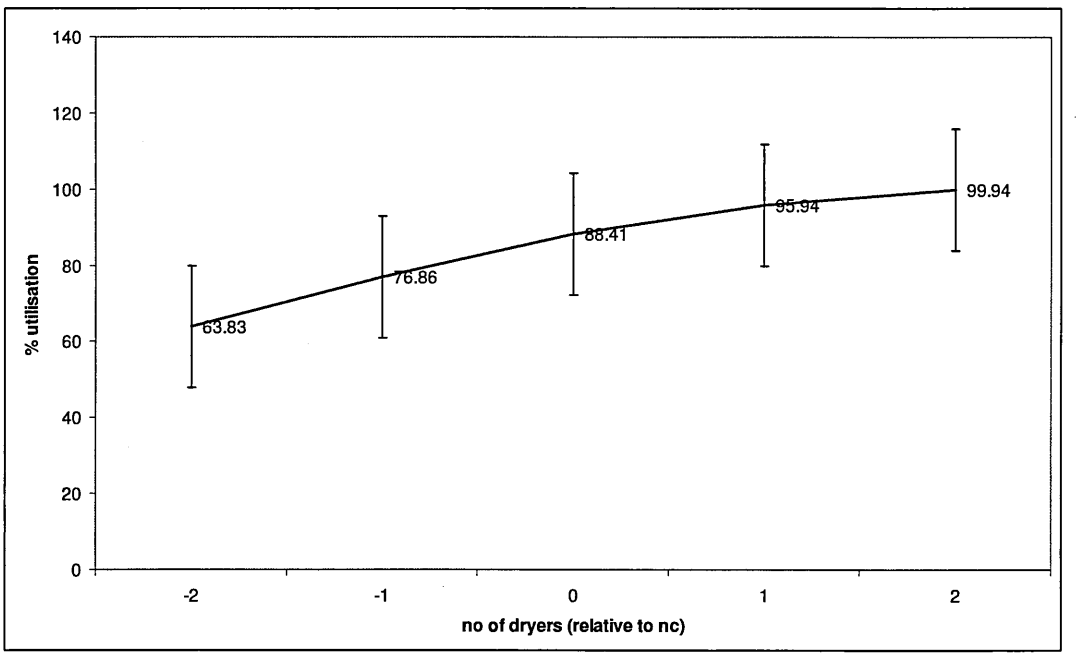


Figure 43: Average results of all experiments

Curve fitting shows that the resultant curve can be accurately modelled within the range shown, using the following formula, and shown in Figure 44 below:

$$y := -.17\,x^3 - 1.58 \cdot x^2 + 9.71 \cdot x + 88.15$$

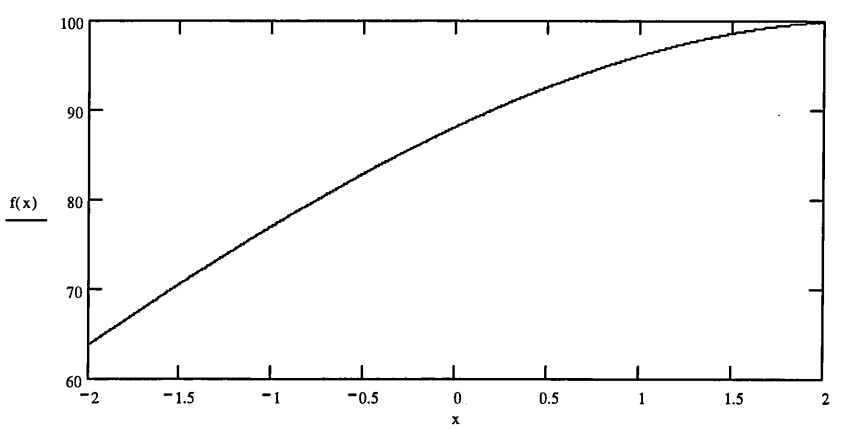


Figure 44: Curve fitting for all average results

This is included for completeness and there is no reason to consider that this formula can be used to predict the behaviour of this system outside of the range -2 < +2 but it is proposed that this polynomial could be used in a future development of the capacity calculation as it is simple to implement.

8.4 Conclusion

This gives the indication that if the conventionally calculated number of dryers is installed, then the system output will be 88% of maximum. One additional dryer will achieve 96% of maximum, and two additional dryers will achieve approximately 100%. This clearly shows the limitation of the conventional calculation, and the benefit of using this simulation approach. This also gives credence to the industry norm rule-of-thumb of achieving 85-90% of maximum output.

This experiment has produced these measurements. It has not shown, nor was it designed to show, explicitly why this should be the case. However considering previous work regarding linked sequential systems, for example, (Goldratt, 1984) it has been shown that time is lost due to the imperfect synchronisation between machines and that any time lost due to perturbations from perfect flow can never be regained. Thus any linked system will produce less than its theoretical maximum. The conventional calculation here gives the theoretical maximum and does not consider the imperfections in the system. The simulation is closer to the reality and therefore yields a more realistic result.

The conclusion relevant to the field of industrial laundry systems and linked sequential systems more generally, is that this ABM methodology gives a more realistic performance model than the conventional calculation, and therefore

there is significant benefit in using this approach in planning and specifying such systems, and furthermore this approach can be used to provide evidence to support management decision making (for example – scenario based planning) in a manner that is not possible with the conventional means of analysing systems.

Chapter Nine: Verification and Validation of the Agent Based Model

Following the work described in the previous chapter, a comparative exercise was carried out on a laundry site in Dunstable, UK, to test the operation of the model.

The site was visited on Thursday 25 August 2011, and grateful thanks are made to Mr Paul Janes, the Production Manager, for providing key data. Production data was provided for 8 weeks commencing 06 June 2011, which was considered to be representative of the normal performance of the plant.

The verification of the ABM was done in general accordance with the scheme outlined in (Niazi, et al., 2009), which was:

1. **verify** the model – debugging. It is notable that this is relatively straightforward with ABM because the microscopic behaviour of each agent is generally simple and well understood.
2. **validate** the model – compare the overall performance of the model with the real world situation, in three ways:
    a. validate using animations – a Subject Matter Expert (SME) to observe the ABM performance and look for violations.
    b. validate using logs – the performance of a simulation can be examined in retrospect.
    c. validation using invariants – the SME to set notifications so when any particular condition is violated the simulation would report this.

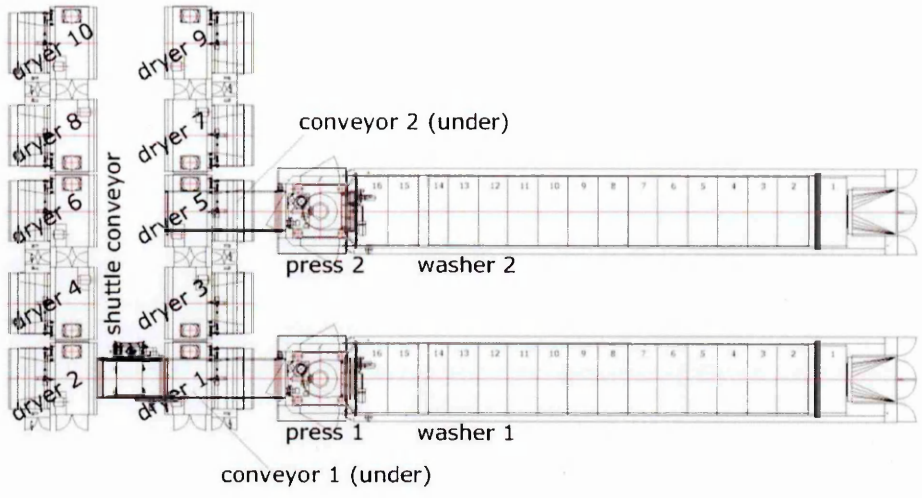
Note that some of these validations can happen at the same time as the simulation is running.

9.1 General Background

The site chosen is operated by Synergy Healthcare PLC who are a major service company in the Healthcare (Hospital) linen sector, with approximately 22% UK market share. The Dunstable site operates 114 hours per week (over 6.5 days) and processes typically 830,000 pieces of laundry (which is about 400,000kg).

Figure 45 below gives a schematic layout of the washing plant, which comprises two batch washing machines, each with a press, an intermediate storage conveyor and a shuttle conveyor taking work to one of 10 dryers.

Not shown in the figure is the overhead bag system feeding work into the washers, or the unloading conveyors which take work away from the dryers into another bag storage and distribution system.



-

Figure 45: General layout for system modelled

Figure 46 shows the central zone between the dryers, with the shuttle conveyor lifted to the loading position for two dryers.

Figure 46: Central zone and shuttle conveyor betweeen dryers

9.1.1 Wash System Performance

Table 25 below gives key overall performance data for the plant (provided by site management). Pieces, wash-loads, and total weight produced, is given for 8 separate weeks in 2011.

week commencing	06-Jun	13-Jun	20-Jun	27-Jun	04-Jul	11-Jul	18-Jul	25-Jul
total pieces produced	833,986	833,027	837,265	833,840	813,177	833,770	824,712	817,513
total loads produced								
washer 1	2,868	2,864	2,676	2,725	2,829	2,677	2,857	2,865
washer 2	2,610	2,504	2,601	2,475	2,618	2,642	2,745	2,611
total	5,478	5,368	5,277	5,200	5,447	5,319	5,602	5,476
total weight	410,850	402,600	395,775	390,000	408,525	398,925	420,150	410,700
average piece weight /kg	0.493	0.483	0.473	0.468	0.502	0.478	0.509	0.502
				average pieces per week	828,411	note 1		
				average total loads produced per week	5,396	note 2		
				average total weight /kg	404,691	note 3		
				average piece weight /kg	0.489	note 4		
working 114 hours per week								
				average weight per hour	3550	kg		
				average loads per hour washer 1	24.52			
				average loads per hour washer 2	22.81			
				average loads per hour total	47.33			

Table 25: Overall performance data for the plant

note 1 - average is the simple mean

note 2 - taken directly from wash control computer

note 3 - each load is programmed to be 75kg and the working assumption was that this was achieved +/- 2kg on each load

note 4 - industry norm for estimation purposes if that a piece = 0.5kg and this estimation is given credence with this data

With reference to terms and general background from chapter 3. The control system defined cycle time of each washer is set to 100 seconds, and together with 4 seconds dwell time between loads (manually timed) this gives the total cycle time as 104 seconds. This represents the load-load 'Total Cycle Time' which is synonymous with the parameter 'Cycle Time' in chapter 3 section 3.3.

Equation 1 from section 3.3 gives the maximum theoretical production rate of each continuous tunnel washer:

L load size (individual batch size - here consistently 75kg)

I_o overload factor (here zero)

C cycle time (104 seconds)

U utilisation factor (approximate actual production as a % of maximum)

$$P_{max} := \frac{3600}{C} \cdot L \cdot (1 + i_o) \qquad \text{equation 1}$$

P_{max} = 34.6 loads per hour * 75kg loads

= 2596 kg per hour

Two washers would therefore produce a maximum of 69.2 loads per hour. From section 3.3, actual expected production:

$$P_{av} := \frac{3600}{C} \cdot L \cdot U$$ equation 2

As $P_{av} = 47.33$ from Table 25 above

$$U := \frac{P_{av}}{P_{max}}$$

U = 68.4%

Working out similar data for the best productive week, that commencing 25 July 2011, provides the following results:

total pieces produced		817,513
total loads produced		5476
average loads produced per hour		48.04
U = 48.04 / 69.2	=	69.4%

9.1.2 Best Production

Snapshots from the production data for one washer, for the immediate days preceding the visit were:

- Monday 22 August 2011, from 0800-1430 147 loads in 6.5 hours
- Tuesday 23 August 2011, from 0800-1430 164 loads in 6.5 hours
- Wed 24 August 2011, from 0800-1430 178 loads in 6.5 hours

These figures related to washer number 1, which had the better production data.

The best figure was 178 loads, equivalent to 27.4 loads per hour. As one washer could produce a maximum of 34.6 loads per hour, this equates to 79.2% utilisation

9.1.3 Reasons

Anecdotally, the plant management gave the following reasons for achieving less than optimum performance, roughly in order of significance:

1. Work delivery to the plant is not perfectly consistent.
2. Over the course of each week there are significant times when the plant simply runs out of work and the system is shut down. This is estimated to cause 3-5 hours stoppage each week.
3. Work mix is not perfectly homogeneous. The washers call off work according to a predetermined sequence table (for example - 3 loads sheets, then 2 loads towels, then 1 load pillowcases etc). The sequence table is set up to be suitable for the work incoming, the work required, and so that the wash (dry) system can process this efficiently. However, if a type of work is not available then the call off steps forward to the next entry in the sequence and the actual sequence may therefore not be optimum for the system.
4. Drying capacity is not quite sufficient for the needs, and sometimes holds up the washing system if all dryers are full. (This is exacerbated by and related to point 2 above.)
5. Typically each week there would be 1-2 major breakdowns causing stoppages of 30-60 minutes.
6. Minor breakdowns or maintenance periods causing shorter stoppages.

9.2 Dryer Parameters

The plant uses 11 wash programme, with dryer programme times as shown in Table 26 below.

type	sheets	pillowcase	towel	blanket	drawsheet	duvet	counterpane	bed cover	patient gown	scrub suit	nightwear
programme	1	2	3	4	5	6	7	8	9	10	11
loading	10	10	10	10	10	10	10	10	10	10	10
drying	55	180	870	1020	360	420	300	660	780	240	240
cooldown	1	1	120	120	1	1	1	120	120	1	1
unloading	35	45	40	45	40	35	35	35	40	40	40
lint cycles	5	5	3	3	2	2	2	2	3	2	2
lint time per cycle	24	24	40	40	60	60	60	60	40	60	60
total dry time	130	265	1083	1238	473	528	408	887	993	353	353
(min)	2.17	4.42	18.05	20.63	7.88	8.80	6.80	14.78	16.55	5.88	5.88

Table 26: Dryer programmes

(all times in seconds except where indicated)

Note: the lines referring to 'lint' relate to the lint cleaning programme which runs after a programmable number of cycles, taking 2 minutes each time. For example, when running programme 1 (sheets) the lint cleaning routine will run every 5th load, therefore 120s/5 = 24 seconds are allocated to each sheet drying programme.

The following data were obtained by manually timing the system operations:

load LSC	6
load dryer	6
unload dryer	0

('unload dryer' is given a zero figure, as this figure was actually found to vary by load and has therefore been included in the drying time - see Table 26)

The shuttle travel times were obtained by manually measuring the system (figures were rounded to the nearest second). It was found in fact (to the accuracy of measurement) that the system was symmetrical - travel from position X to Y was the same for travel from position Y to X - although this is not necessarily expected to be the case for all systems. The travel times are recorded in

Table 27 below:

	timings	start 1	2	3	4	5	6	7	8	9	10	11	12
		W1	W2	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
dest 1	W1	0	4	11	11	12	12	16	16	17	17	19	19
2	W2	4	0	16	16	12	12	11	11	12	12	16	16
3	D1	11	16	0	0	4	4	6	6	8	8	9	9
4	D2	11	16	0	0	4	4	6	6	8	8	9	9
5	D3	12	12	4	4	0	0	4	4	6	6	8	8
6	D4	12	12	4	4	0	0	4	4	6	6	8	8
7	D5	16	11	6	6	4	4	0	0	4	4	6	6
8	D6	16	11	6	6	4	4	0	0	4	4	6	6
9	D7	17	12	8	8	6	6	4	4	0	0	4	4
10	D8	17	12	8	8	6	6	4	4	0	0	4	4
11	D9	19	16	9	9	8	8	6	6	4	4	0	0
12	D10	19	16	9	9	8	8	6	6	4	4	0	0

Table 27: Shuttle travel times /s, lookup table.

9.3 Dry Times

In order to obtain an accurate mix of work, production data was taken from each washer from the last time the washer counters were reset, giving an average over a long period. This data is recorded in Table 28 below.

It should be observed that washer 2 is considerably older than washer 1, hence its production counters extend over a very long period (3-5 years). The mix from both washers were calculated independently as the plant management do tend to prioritise towels through washer 2, and sheets through washer 1. Therefore the work mix through each washer will not be representative of the overall plant work mix.

type / programme	sheets 1	pillowcase 2	towel 3	blanket 4	drawsheet 5	duvet 6	counterpane 7	bed cover 8	patient gown 9	scrub suit 10	nightwear 11
washer 1	2,978	137	268	3,312	967	191	336	1,729	1,183	995	95
% of total	24.43%	1.12%	2.20%	27.17%	7.93%	1.57%	2.76%	14.18%	9.70%	8.16%	0.78%
total 12,191											
washer 2	197,044	22,140	40,501	926	758	147	305	490	366	1,452	7,825
% of total	72.45%	8.14%	14.89%	0.34%	0.28%	0.05%	0.11%	0.18%	0.13%	0.53%	2.88%
total 271,954											
normalised totals											
	48.44%	4.63%	8.55%	13.75%	4.11%	0.81%	1.43%	7.18%	4.92%	4.35%	1.83%

Table 28: Number of loads produced by each machine for each programme

(non-percentage numbers refer to numbers of loads produced)

The different work programmes were then split into the normal three categories (used for system design as outlined in section 3.3) and the average dry time for each category was calculated pro-rata according to the sub-categories (see Table 29).

major category	programme code	type	dry time (/s)	% of mix	% of category
no dry		**category total**	**130**	**48.44%**	
	1	sheets	130	48.44%	100.00%
medium dry			**371**	**17.16%**	
	2	pillowcases	265	4.63%	26.98%
	5	drawsheet	473	4.11%	23.95%
	6	duvet	528	0.81%	4.72%
	7	counterpane	408	1.43%	8.33%
	10	scrub suit	353	4.35%	25.35%
	11	nightwear	353	1.83%	10.66%
full dry			**1091**	**34.40%**	
	3	towel	1083	8.55%	24.85%
	4	blanket	1238	13.75%	39.97%
	8	bed cover	887	7.18%	20.87%
	9	patient gown	993	4.92%	14.30%

Table 29: Calculation of dry times for each of the three design categories

Note: The % of category column was used to calculate the average dry time for each category on a weighted basis.

From Table 29, the following data can be extracted to input into the Agent Based Model:

	%	code	time
no dry	48.44	1	130
med dry	17.16	2	371
full dry	34.40	3	1091

Table 30: Average dry times for the three basic categories

9.4 Static Calculation

The results of carrying out the static calculation (as per the procedure outlined in section 3.4) are shown in Table 31.

	%	code	dry-time	total d-t	loads/hr	dryer-min/hr	dryers	total
no dry	48.44	1	130	136	33.5353	76.0	1.27	
med dry	17.16	2	371	377	11.88	74.6	1.24	**9.77**
full dry	34.4	3	1091	1097	23.8153	435.4	7.26	

Table 31: Static calculation for number of dryers required

In total, 9.77 dryers are required. The standard estimation based on this static calculation would be that 10 dryers would be installed. However the conclusion of chapter 8 gives reason to conclude that this would only lead to 88% of maximum

capacity being produced. In fact, the reasons quoted in section 9.2 lead to a lower system utilisation than that, although the fact that 'lack of dryer capacity' is quoted as one reason is given credibility here.

9.5 Agent Based Model Calculation

With the collected data entered into the model, the model was run 14 times for a simulated 4 hours each time.

parameter	name	type	range / notes
n_c	number of dryers	fixed	$n_c = 9.77$ (according to section 9.5 above)
n	number of dryers (integer)	variable	n = 10 (existing situation on site)
c	washer cycle time	fixed	104 (from site)
x:y:z	work mix	fixed	48.44% : 17.16% : 34.40% (from section 9.4 above)
tx ty tz	dry time (x/y/z)	fixed	tx = 130 ty = 371 tz = 1091

			(from section 9.4 above)
LLSC	load LSC	fixed	6 (from site)
LD	load dryer	fixed	6 (from site)
UD	unload dryer	fixed	0 (as per section 9.3 above)
T(14,14)	travel times	fixed	as Table 27 above

Table 32: Parameters for each run

The outputs of each run were:

parameter	name	type	range / notes
LP	number of loads produced	integer	for 4 hours production and c=104 the theoretical max is 277. LP=total of loads produced by each dryer.
LP%	number of loads produced %	%	LP / 277 * 100%
x':y':z'	work mix	3 x rounded to 2dp	% work mix such that x' + y' + z' = 100%
m	number of dryers based on x'		if the actual work mix is used as the input to the conventional dryer calculation, m will differ from n.

Table 33: Outputs for each run

9.6 Results obtained

Table 33 below provides all results obtained for all 14 runs.

index	LP	LP%	x'	y'	z'	n(c)	m	m / nc	LP% * m/nc
1	275	99.31%	53.45%	18.55%	28.00%	9.77	8.65	0.89	87.93%
2	272	98.22%	53.31%	18.38%	28.31%	9.77	8.70	0.89	87.47%
3	263	94.97%	50.19%	18.63%	31.18%	9.77	9.24	0.95	89.85%
4	268	96.78%	52.61%	22.01%	25.37%	9.77	8.32	0.85	82.48%
5	271	97.86%	53.14%	18.82%	28.04%	9.77	8.67	0.89	86.86%
6	263	94.97%	50.19%	18.63%	31.18%	9.77	9.24	0.95	89.85%
7	268	96.78%	52.61%	22.01%	25.37%	9.77	8.32	0.85	82.48%
8	245	88.47%	49.80%	13.47%	36.73%	9.77	10.03	1.03	90.83%
9	254	91.72%	48.43%	12.60%	38.98%	9.77	10.40	1.06	97.68%
10	265	95.69%	50.57%	17.74%	31.70%	9.77	9.30	0.95	91.06%
11	248	89.56%	45.56%	17.74%	36.69%	9.77	10.22	1.05	93.69%
12	252	91.00%	48.02%	15.08%	36.90%	9.77	10.13	1.04	94.41%
13	264	95.33%	48.48%	17.05%	34.47%	9.77	9.78	1.00	95.41%
14	263	94.97%	50.95%	17.87%	31.18%	9.77	9.21	0.94	89.50%
average	262.2	94.69%	50.52%	17.76%	31.72%	9.77	9.30	0.95	89.96%

Table 34: Summary all results obtained

On average therefore the ABM calculated that the system with the dryers as specified should produce approximately 90% its design maximum. However, from section 9.1 it was found that the average real production was below this being 69% on average and 79% best. The reasons given in section 9.2.2 would explain this variance.

9.7 Conclusion

This exercise has verified and validated the ABM simulation, and shown that it is effective at modelling the real world system more accurately than the previous static calculation.

Although the system utilisation figures are generally good (ranging from 86 - 98% for the figures corrected by the m/n_c factor), it is noticeable that for some experimental runs, the work mix (which is stochastic within the probability function given by x:y:z) sometimes varies thereby requiring more than 10 dryers on the conventional calculation (and this is where the laundry find that the dryer capacity is not sufficient). The m/n_c factor provides a good measure of the variation of work mix from the planned specification.

This exercise has also given good evidence that other factors (for example, work being available to wash) affect the system utilisation a good deal more than the dryer capacity.

The benefit of the ABM simulation is that it is able to predict and quantify the system utilisation better than the previous static model. However, given the additional factors (section 9.2.2) the use of the ABM must still be made with care, and some of these other factors would affect the productivity of the plant and may not be able to be predicted or modelled.

Chapter Ten: Design Exercise

In order to prove the usefulness and operation of the Agent Based Model, a design exercise was conducted. Thanks are due to Kannegiesser UK Ltd. who provided the input data from a new commercial laundry facility in Cardiff. (Afonwen Services Ltd).

10.1 Installed Plant Design

For illustration, Figure 47 below gives the overall system design as installed.

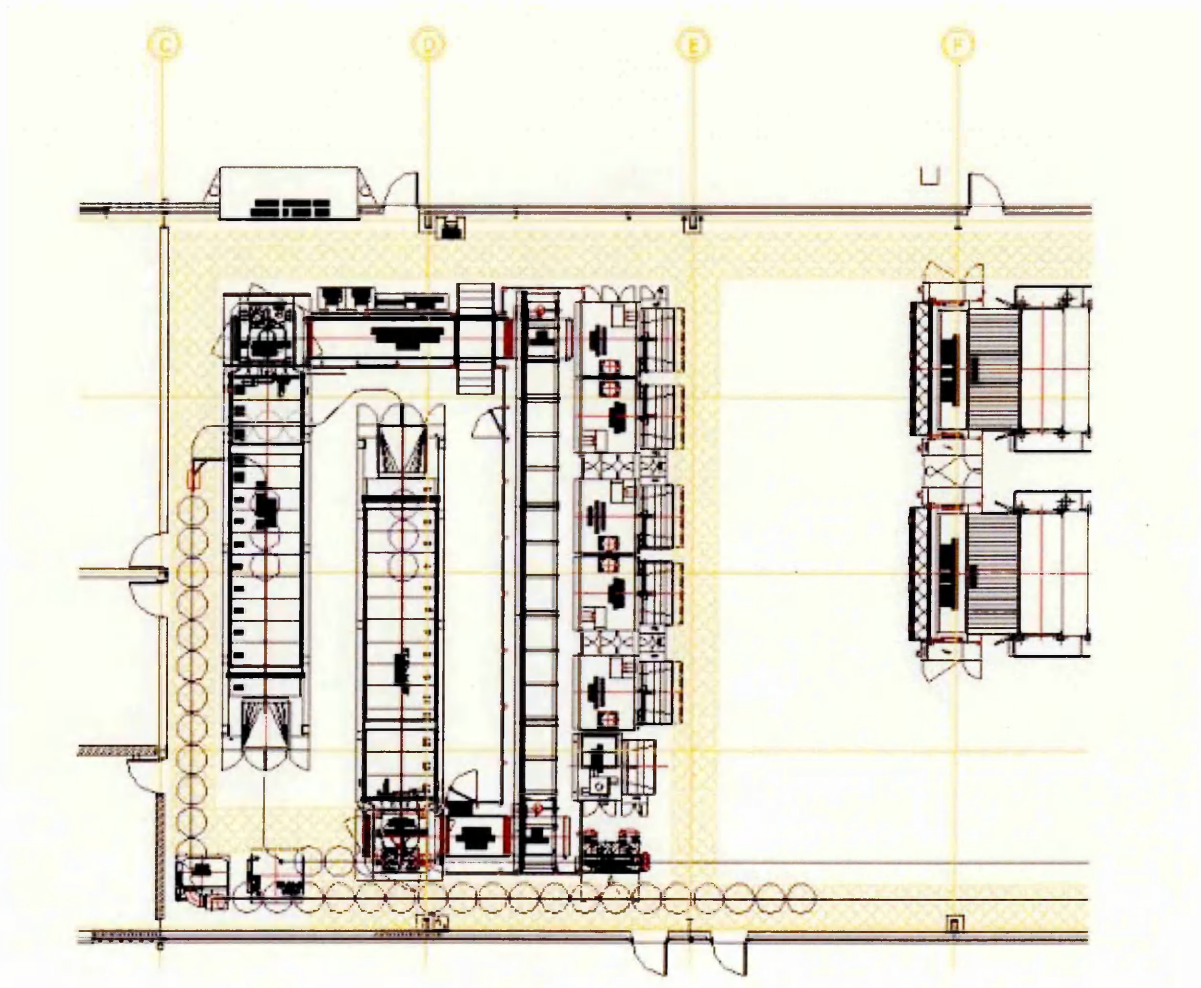


Figure 47: Overall Cardiff plant design

There are 6 dryers shown. The 5 larger ones take double loads from the washers and therefore are equivalent to the 10 required from the conventional calculation. The smaller one is the special version supplied without heating for the conditioning only of no-dry work.

10.2 Design Parameters

The system was specified in the manner outlined in section 3.2. Table 35 shows the main production requirements.

	150,000	net pieces per week			
	40	hours run per week			
		work breakdown			
	% by pieces	average weight (kg)	dry code	net pieces per hour	net weight per hour
total	**100.0%**	**0.500**		**3,750**	**1,875.4**
sheets	13.00%	0.580	none	488	282.8
duvets	5.00%	0.780	part	188	146.3
pillowcases	24.00%	0.180	part	900	162.0
kitchen cloths	8.00%	0.250	part	300	75.0
napkins	10.00%	0.100	none	375	37.5
table linen 36"	1.00%	0.300	none	38	11.3
table linen, other	3.00%	0.600	none	113	67.5
table linen large	1.00%	0.800	none	38	30.0
towels	28.00%	0.480	full	1,050	504.0
mats	7.00%	2.130	full	263	559.1

Table 35: Key input data, production requirement and work mix

10.3 Conventional Calculation

10.3.1 Calculation of Washers

Following the process of calculating the wash system, as outlined in section 3.3.1, with key data drawn from Table 35:

- decide on amount of work per hour to produce
    - 150,000 pieces to be processed in a 40 hour shift. With typical piece weights this equates to 1875kg/hour.
- decide on best batch size
    - due to customer trends, 50kg was considered the right load.
- 1875kg per hour / 50kg batch = 37.5 batches per hour.
- based on a typical utilisation factor of 85% the system should be sized to produce a maximum of 37.5 / 0.85 = 44.1 batches per hour (in order to actually output an average of the required production).
- 44 batches is not possible for a single washing machine, so two washers were specified – each to produce a minimum of 22 batches per hour.
- To produce a required 22 batches per hour, the maximum possible cycle time is 3600 / 22 = 163s. In fact 150s was chosen to give some overloading security.
- Decide on the required maximum wash time
    - customer requirement was for very high quality, therefore a wash time of over 30 minutes was chosen. 14 compartments were chosen with a cycle time of 150s gives a wash time of 35 minutes.

Therefore for this project, two 50kg batch size machines each with 14 compartments, washing at 150s cycle time, were chosen. This gives a total wash capacity of 2040kg/hour, which is 165kg/hour, (8.8%) over the requirement.

10.3.2 Calculation of Dryers

Following the procedure outlined in section 3.4, and with key data drawn from Table 35, the calculation is summarised in

Table 36 showing that 10 dryers are required.

batches/hour to dry	48.0 (worst case)		
	for each dry code		
	none	part	full
dry time (min)	0	8	18
% of work	22.9%	20.4%	56.7%
batches/hour	11	10	27
dryer-min	0.0	78.5	489.6
dryers	0.0	1.3	8.2
total dryers required			10

Table 36: Work mix for drying

In fact due to space restrictions, and the nature of the customer base where the work was able to be grouped into very large loads, it was decided to use larger dryers each of which could take two loads at the same time (double batching). Therefore the number of dryers effectively halves – only 5 are required.

Note that the no-dry category has been excluded – with a zero dry-time – because it was decided for cost reasons to install one dryer without heating (which would do all the no-dry categories simply loosening up the batch prior to further processing). Therefore, the installation requirement was 5 double batch dryers *plus* the no-heating shaker-dryer.

10.4 ABM Calculation

The starting point is the output of the conventional calculation given above. The outputs of the ABM are given in Table 37.

n	LP	LP%	x'	y'	z'	n(c)	m	total waiting time /min	% of total	m / nc	LP% * m/nc
2	77	40.10%	20.78%	25.97%	53.25%	4.86	4.78	286.08	59.60%	0.98	39.50%
3	122	63.54%	25.41%	21.31%	53.28%	4.86	4.64	178.62	37.21%	0.95	60.67%
4	154	80.21%	19.48%	25.97%	54.55%	4.86	4.88	101.15	21.07%	1.00	80.56%
5	183	95.31%	19.67%	24.59%	55.74%	4.86	4.92	15.40	3.21%	1.01	96.56%
6	193	100.52%	19.69%	25.91%	54.40%	4.86	4.87	0.00	0.00%	1.00	100.71%

Table 37: ABM results

In the table, n shows the actual number of dryers the simulated system is designed with, and LP the loads produced by that system. Clearly a system with more dryers would be expected to produce more loads, and this is what is found. The ratio x':y':z' – the actual work ratio - is shown with 'm' - the number of dryers that would be required for this work ratio by the conventional calculation. The ratio m/n_c is then used to normalise the results. The final column shows the loads produced, as a percentage of maximum, normalised by this ratio. This then gives a good measure of the utilisation of the washing system.

The column 'total waiting time' also shows how much time was wasted because the system was held up waiting for a dryer to become free. This is also a good measure of the utilisation of the system.

This shows that with 5 dryers (n=5), the loads produced would be expected to be approximately 95% of the maximum possible. This is consistent with reports back from the site where they found that the dryer capacity installed was not quite enough, although this was not quantified.

These results also show that if 6 dryers were to be installed, the dryer capacity would no longer be a restriction on the washing system capacity.

Note that where the system production exceeds 100%, this is due to the exact timing of loads – a batch is counted when it exits a dryer, and the timing of this may mean that a previously processed batch may just fall into the counting time for the experiment.

10.5 Comparison and Conclusion

The dryer capacity of this plant was originally calculated and specified using the conventional static model. It has been found by running the ABM simulation that the dryer capacity is in fact not quite enough (95% capacity) and this is consistent with informal reports from the site.

It is concluded therefore that the ABM gives a better understanding of the performance of the system than the static model, and had this been used prior to installation, a slightly increased dryer capacity would have been proposed which would have improved the subsequent performance of the plant.

Chapter Eleven: Conclusions and Recommendations

This chapter draws conclusions to the entirely of the research, with proposals for further research, consideration of reflective practice, and puts forward implications of the research.

This research has led to a new way of analysing the dynamics of linked sequential systems in industrial laundries. This development of the Agent Based Dynamic Model simulation is novel in the field of industrial laundry, and is a marked improvement on the current static approach. The model is a new and effective method of understanding and predicting the performance of such systems, and offers significant advantage over the current methods.

The mathematical analysis of the calculation of performance of both tunnel washers and dryer systems presented here (sections 3.3 and 3.4 respectively) is developed in a more rigorous manner than any previous literature.

The research into application of AI methods to adaptive controls has added to existing scholarship in these methods and contributed to the overall depth of knowledge: specifically by application to different situations not previously tried. Also, the method of combination and application of EAs and ANNs is novel.

11.1 Conclusions to the Research Questions

The research questions (from section 1.2) were:

Research Question 1. (repeated from section 1.2).

Is it viable to apply the AI paradigms of Evolutionary Algorithms (EAs) and Artificial Neural Networks (ANNs) to the control of linked sequential systems in industrial laundries, in particular at a decision making point?

If so, can specific strategies be identified for their implementation, in order to develop better methods for such control and thereby improve the performance of such systems?

Research Question 2. (repeated from section 1.2).

Does Agent Based Modelling offer an effective approach to the simulation of linked sequential systems in industrial laundries?

If so, is this a better method of simulating such systems than the existing methods, and can it offer a better method of predicting the performance of such systems in order to improve the specification of such systems at the design stage?

11.1.1 Conclusion to Research Question 1

It has been found that it is viable to apply EANNs to provide the decision making process in the control of linked sequential systems in industrial laundries.

This conclusion is supported by the work presented in chapters 4-7. In particular, chapter 4 showed with an abstract example that EANNs can be used to classify, using only proportional information provided about a population. Meanwhile

chapter 5 showed that in general terms, this EANN approach can be used to generate ANNs which can provide effective decision making at the point investigated, with only overall production information provided as feedback.

Chapter 6 then showed that there are limitations to this approach, reporting that if an effective solution is not found then it is impossible to say without further testing if this is a problem with the EA or the ANN, and therefore how to resolve this.

Chapter 7 investigated the use of more sophisticated parameters in the use of EANNs. Further work would be required to determine more explicitly the best use of these in future applications.

11.1.2 Conclusion to Research Question 2

It has been found that ABM can be used to improve the modelling of linked systems, and this can be done during system design, to better specify the system required. It has been shown in chapter 10 that the ABM simulation developed during this research provides a more accurate and more effective method of predicting the performance of such systems. It was also shown that the ABM simulation can be used at the design stage for specification purposes, or for an installed system in support of management decision making. This conclusion is supported by the work presented throughout chapters 8-10.

11.2 General Conclusions for Control and Simulation of Linked Systems

It was found that the dynamics of Linked Sequential Systems in the context of industrial laundry were not well understood and calculation of these systems tended to be based on simple static calculations.

In relation to ABM simulation of production systems, this research builds on the work by (Barbosa & Leitao, 2011) and (Parunak, et al., 1998), and adds to the literature in this field.

It has been found that ABM can be an effective system of modelling, in particular because:

1. the ABM is built from small agents, which can be generally described by simple behaviour, so they can be specified and implemented with confidence.
2. the ABM is dynamic and the complexity of the behaviour of the whole system is an emergent property.
3. the ABM can be verified and validated in a number of ways some of which can occur in parallel to the modelling.
4. because (i) the essence of the ABM is in the interfaces between the agents, (ii) the agents generally pass only limited or simple information between each other, and (iii) the information that is passed is usually fixed regardless of how the agent is modelled, there is no reason why an ABM cannot be updated on an agent by agent basis, as agents are changed or

the modelling of a particular agent can be made more accurate. This is different from a centralised equation based approach.

5. ABM allows for an easy way of simulating scenarios, for either management decision making support, or design specifications.

11.3 Further Research

There are two main avenues of worthwhile further research generated by this work:

1. Further development of the application of EANN methods to controls

More testing of the evolutionary process would enable a better understanding of the most effective parameters for actual implementation, for example, number of generations, mutation rate, plus the more sophisticated parameters investigated in chapter 7 – such as the different payoff functions.

Experimentation could be carried out with changing circumstances to establish the actual adaptation response rate of the system.

2. Further evaluation and development of the ABM simulation

In order to further increase reliability and credibility of the model, more comparative work along the lines of the work described in chapter 9 could be carried out. In particular, further comparisons with systems having different specifications and parameters would test the wider limits of the simulation.

Because the essence of an ABM simulation is that the components can be relatively easily modelled, the overall simulation can be built up step by step and more components added in. Additionally existing components can be modelled differently, either because their operation can be modelled in a more sophisticated way, or because they change, or because the effect of a proposed change is needed to be modelled and tested. The ABM could be developed further to take into account more components with a view to simulating the operation of the overall production facility (not just a single production line).

The ABM simulation approach in this research has been applied only to linked sequential systems. Of course such systems are operated in the context of a working laundry. It would be worthwhile further work to expand the model in order to incorporate organisational aspects of the overall laundry operation, including the behaviours of the people (individually or as departments) involved. This would lead to a deeper understanding of the overall operation and allow a wider range of scenario based simulations in order to optimise the overall laundry operation including organisational aspects as well as technical.

The ABM simulation has in this research been applied to only industrial laundry systems. There is no reason why the findings here could not be equally applied to other fields – as has been discussed in previous chapters most industrial systems comprise (all or in part) linked sequential systems and so this research work would have valuable application to other fields. In particular, this research gives rise to questions of scaling, such as:

- laundry systems generally comprise one, two or three washing machines, and between four and twenty dryers. How would this approach scale up to

other forms of system where there may be 100s or 1000s of parallel and sequential machines? For example, in the telecommunications industry there are situations where data packets are switched by many servers, and queuing theory has been developed to understand this. How would the ABM methods described here apply to that, and would this be a useful augmentation of queuing theory?

- could this approach be used to model at a different scale – not numbers of units in the system, but sizes of unit? For example – instead of modelling machines within a factory, would it be credible and useful to model factories as units? Most manufacturing now takes place in several factories, with a supply chain, and a supply chain could be modelled as a linked sequential system. Would this approach be effective for this type of system?

11.4 Reflections

As the purpose of a PhD is to provide training into the practice of independent unsupervised research, it is useful to reflect on the process and the lessons learnt. While working on this research I have undertaken the following tasks which have led to a better understanding of the process of research:

- literature search - I have undertaken literature searching both physically and online, using academic journals, books, etc.
- independent and collaborative working - generally the practice of an Open University part-time PhD will involve independent work. However, no

scholarship happens in isolation, and I have learned the importance of the academic community. Contacts made by attending academic conferences have been instrumental, and chance conversations with other researchers have sometimes sparked off ideas and avenues to progress which would not otherwise have happened. I have also worked within the Open University community, within a programme of research of my Supervisors, and occasional seminars meeting other research students have also helped shape the work. Supervision meetings have led to new directions to explore and it is indicative of this that papers have been published both as an individual and also crediting others where this was appropriate due to their input.

- experimental design - I have learned and confirmed the importance of setting out, in advance, the aims and goals of a particular experiment if the work is not to progress in a vague manner. Without clear goals it is likely that the experiment will never in fact end, as there is always the potential to do a little (or a lot) more.

Bibliography

Angeline, P., Saunders, G. & Pollack, J., 1994. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks,* 5(1), pp. 54-65.

Banzhaf, W., Nordin, P., Keller, R. & Francone, F., 1998. *Genetic Programming - An Introduction.* s.l.:s.n.

Barbosa, J. & Leitao, P., 2011. Simulation of Multi-agent Manufacturing Systems using Agent-based Modelling Platforms. *IEEE.*

Barricelli, N., 1962. Numerical testing of evolution theories: Part 1 Theoretical introduction and basic tests. *Acta Biotheoretica,* 16(1-2), pp. 69-98.

Beggs, B., 2006. Tunnel or washer-extractors: whats's the proper choice?. *American Laundry News*, 29 July.

Bremermann, H., 1962. Optimisation through evolution and recombination. In: M. Yovitts, ed. *Self-Organising Systems.* Washington: Spartan Books, pp. 93-106.

Cho, T., Conners, R. & Araman, P., 1991. Fast back-propagation learning using steep activation funtions and automatic weight reinitialisation. *IEEE International Conference, Systems, Man, and Cybernetics,* Volume 3, pp. 1587-1592.

Cohen, I., 2003. *Semisupervised learning of classifiers with Application to Human-Computer Interaction.* s.l.:PhD Thesis.

Curran, D. & O'Riordan, C., 2004. Evolving Connect-Four Playing Neural Networks using Cultural Learning. *http://ww2.it.nuigalway.ie/cirg/publications.html.*

deJesus & Hagan, 2007. Backpropagation Algorithms for a broad class of dynamic networks. *IEEE transactions on neural networks,* 18(1).

Fahlman, S., 1988. *An empirical study of learning speed in backpropagation networks,* s.l.: Technical Report CMU-CS-88-162, Carnegie-Mellon University.

Fahlman, S. & Libiere, C., 1991. *The Cascade-Correlation Learning Architecture.* Pittsburgh: Carnegie Mellon University.

Ferron, Y., 2011. Laundry Upgrade. *Ashrae Journal,* pp. 44-50.

Fogel, D., 1966. *Artificial Intelligence Through Simulated Evolution.* s.l.:s.n.

Fogel, D., 1993. Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe. *Proc. Amer. Power Conf.,* pp. 875-879.

Forrester, J., 1961. *Industrial Dynamics.* Oregon: Productivity Press.

Garey, M., 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research,* 1(2), pp. 117-129.

Goldratt, E., 1984. *The Goal.* s.l.:North River Press.

Hebb, D., 1949. *The Organisation of Behaviour (p62).* New York: Wiley.

Hecht-Nielsen, R., 1989. *Neurocomputing.* s.l.:Addison-Wesley.

Holland, J., 1975. *Adaptation in Natural and Artificial Systems.* s.l.:MIT Press.

Hopfield, J. & Tank, D., 1982. *Neural Networks and physical systems with emergent collective computational abilities.* USA, s.n., pp. 2554-2558.

Hopgood, A., 2000. *Intelligent Systems for Engineers and Scientists.* 2 ed. s.l.:CRC Press.

Hopgood, A. et al., 2004. optimisation of plasma etch processes using evolutionary search methods with in situ diagnostics. *IoP.*

Hornik, K., Stinchcombe, M. & White, H., 1989. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks 2,* pp. 359-366.

Jennings, N., 1996. Distributed Artificial Intelligence Control of Electricity Distribution - ARCHON (ARchitecture for Cooperative Hetergeneous ON-Line systems).

Jennings, N., Wooldridge, M. & Sycara, K., 1998. A Roadmap of Agent Research & Development. In: *Autonomous Agents and Multi-Agent Systems.* The Netherlands: Kluwer Academic Publishers, pp. 275-306.

Johnson, J. & Rose, V., 2005. Shape Recognition using Randomly Selected Pixel Pair Neurons. *Open University.*

Kannegiesser GmbH, n.d. *Calculation of Laundry Capacity,* s.l.: s.n.

Khuri, s. & Miryala, S., 1999. Genetic Algorithms for solving open shop scheduling problems. *Proc. 9th Portuguese Conference on Artificial Intelligence.*

Kim, D., 2002. Standard and advanced backpropagation models for image processing application in traffic engineering. *Intelligent Transportation Systems*, pp. 199-211.

Kohonen, T., 2001. *Self-Organising Maps.* 2 ed. s.l.:Springer.

Kolmogorov, A., 1957. On the representation of continuous functions of many variables by the superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSR,* Volume 114, pp. 953-956.

Koprinkova-Hristova, P., 2010. Backpropagation through time training of a neuro-fuzzy controller. *International Journal of Neural Systems,* 20(5), pp. 421-428.

Kuo, S.-F., Chen, F.-W., Liao, P.-Y. & Liu, C.-W., 2011. A comparative study on the estimation of evapotranspiration using backpropagation neural network. *Paddy Water Environ*, 25 August, pp. 9:413-424.

Liu, Y., You, Z. & Cao, L., 2006. On the almost periodic solution of generalized Hopfield neural networks with time-varying delays. *Neurocomputing,* Volume 69, pp. 1760-1767.

Manikas, A. & Godfrey, M., 2011. Evolutionary Algorithm Parameter Fitness: An exploratory study. *International Journal of Management an Marketing Research,* 4(3), pp. 35-44.

McCulloch, W. & Pitts, W., 1943. A Logical Calculus of the Ideas Imminent in Nervous Activity. *Bulletin of Mathematical Biophysics,* Volume 5, pp. 115-133.

Minsky, M. & Papert, S., 1969. *Perceptrons: An Introduction to Computational Geometry.* s.l.:MIT Press.

Morley, P., 2003. *An Investigation into Automated Laundry Sorting, MEng Project,* s.l.: Open University.

Morley, P., 2005. *Training a Genetic Algorithm and Neural Network Hybrid Pattern Classifier by Population Statistics.* s.l., AISB Conf. Proc..

Morley, P., 2009. *Evolving Neural Networks to Play Noughts & Crosses.* s.l., ISKE.

Morley, P., 2010. *Application of ANN-GA Hybrid to run a Conveyor Control System.* s.l., AISB.

Nejad, H., Sugimura, N. & Iwamura, K., 2011. Agent-based dynamic integrated process planning and scheduling in flexible manufacturing systems. *International Journal of Production Research,* 49(5), pp. 1373-1389.

Niazi, M., Hussain, A. & Kolberg, M., 2009. Verification &Validation of Agent Based Simulations using the VOMAS (Virtual Overlay Multi-agent System) approach.

Nilsson, F. & Darley, V., 2006. On complex adaptive systems and agent-based modelling for improved decision making in manufacturing and logistics settings. *International Journal of Operations & Production Management,* 26(12), pp. 1351-1373.

Nolfi, S. & Parisi, D., 1997. Learning to Adapt to changing environments by evolving neural networks. *Adaptive Behaviour,* Volume 5.

Palmes, P., Hayasaka, T. & Usui, S., 2005. Mutation Based Genetic Neural Network. *IEEE Transactions on Neural Networks,* 16(3), pp. 587-600.

Panoiu, M., Panoiu, C., Iordan, A. & Illes, C., 2011. Software package for analysis the performances of backpropagation neural networks training algorithm. *Annals of faculty engineering Hunedoara - International Journal of Engineering*, pp. 164-166.

Parunak, H., Savit, R. & Riolo, R., 1998. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. *Proc MABS (Modeling Agent Based Systems).*

Penrose, R., 1989. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics.* s.l.:Penguin Books.

Picton, P., 1994. *Introduction to Neural Networks.* s.l.:Macmillan.

Poli, R. & Langdon, W., 1998. Schema Theory for Genetic Programming with One-Point Crossover & Point Mutation. *Evolutionary Computation,* 6(3), pp. 231-252.

Poli, R. & McPhee, N., 2003. general schema theory for genetic programming with subtree swapping crossover part I. *Evolutionary Comutation,* 11(1), pp. 53-66.

Poli, R., McPhee, N. & Rowe, J., 2004. exact schema theory and markov chain models for genetic programming and variable length genetic algorithms with homologous crossover. pp. 1-38.

Reaidy, J., Liu, Y., Diep, D. & Massotte, P., 2003. Intelligent agents for production systems. In: S. d'Amours & A. Guinet, eds. *Intelligent Agent-Based Operations Management.* London: Kogan Page Ltd, pp. 147-164.

Renna, P., 2011. Multi-agent based scheduling in manufacturing cells in a dynamic environment. *International Journal of Production Research,* 49(5), pp. 1285-1301.

Rogers, P., 2003. Tips to keep tunnels in tune, on time. *American Laundry News,* 12 January.

Ruiz, N., Giret, A., Botti, V. & Feria, V., 2011. Agent-supported simulation environment for intelligent manufacturing and warehouse management systems. *International Journal of Production Research,* 49(5), pp. 1469-1482.

Rumelhart, D. & McClelland, J., 1986. *Parallel Distributed Processing - explorations in the microstructure of cognition.* Cambridge MA: MIT Press.

Schwefel, H., 1977. *Numerical Optimisation of Computer Models.* Basle: Birkhauser.

Sharma, S., Kumar, D. & Kumar, A., 2012. Reliability analysis of complex multi-robotic system using GA and fuzzy methodology. *Applied Soft Computing,* Volume 12, pp. 404-415.

Wang, J., 2006. An improved discrete Hopfield neural network for Max-Cut problems. *Neurocomputing,* Volume 69, pp. 1665-1669.

Whitley, D., Rana, S. & Heckendorn, R., 1998. Island Model Genetic Algorithms & Linearly Separable Problems. *Journal of Computing & Information Technology.*

Widrow, B. & Hoff, M., 1960. Adaptive Switching Circuits. *IRE WESCON Convention Record,* Volume 4, pp. 96-104.

Wisittipanich, W. & Kachitvichyanukul, V., 2012. Two enhanced differential evolution algorithms for job shop scheduling problems. *International Journal of Production Research,* 50(10), pp. 2757-2773.

Wolpert, D. & Macready, W., 1997. No Free Lunch Theorems for Optimisation. *IEEE Transactions on Evolutionary Computation,* 1(1), pp. 67-82.

Wooldridge, M., 1997. Agent-Based Software Engineering. *IEE Proc. Software Engineering,* Volume 144, pp. 26-37.

Yao, X., 1999. Evolving Artificial Neural Networks. *Proc IEEE,* 87(9).

Yogeswaran, M., Ponnambalam, S. & Tiwari, M., 2009. An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS. *International Journal of Production Research,* 47(19), pp. 5421-5448.

Appendices

Appendix A

Previously published paper:

"Training a Genetic Algorithm and Neural Network Hybrid Pattern Classified by Population Statistics". AISB, 2005. (this reports on some of the work described fully in chapter 4)

Appendix B

Previously published paper:

"Application of EANN Hybrid to run a Conveyor Control System". AISB, 2010. (This reports on some of the work described fully in chapter 5)

Appendix C

Previously published paper:

"Evolving Neural Networks To Play Noughts and Crosses". ISKE, 2009. (This reports on some of the work described fully in chapter 7)

Appendix D

Calculation sheet showing the conventional calculation on wash and dryer required capacity for an industrial laundry.

Training a Genetic Algorithm and Neural Network Hybrid Pattern Classifier by Population Statistics

Paul Morley
Department of Design & Innovation
The Open University
Milton Keynes, MK7 6AA, England
pm4958@tutor.open.ac.uk

Abstract

A method of unsupervised learning is proposed that uses a genetic algorithm to train a neural network to implement a classification function where the only a priori information known about the population to be classified is the relative proportions of the different types. The genetic algorithm is used to evolve the weights in a network in order to provide some classification function that provides the same proportional split as that known, and it is postulated that generally this will provide the classification function required. It is furthermore postulated that with a minimal amount of human guidance the performance of the evolution can be directed, in an analogous manner to a human dog breeder who utilises the mechanics of evolution to a conscious end

1. Introduction

Artificial neural networks (ANNs), and genetic algorithms (GAs), are both well known and widely used in artificial intelligence fields, including their application to problems such as pattern recognition and image analysis. ANNs in particular have been shown to be extremely powerful for classification problems - it has been shown (Hornik *et al*, 1989) that in principle a multi-layered perceptron can approximate any classification function.

The problem with complex ANNs, is finding the weights of the links. Various methods have been tried, mostly stemming from the delta rule; where the change in the link weight is proportional to the difference between the desired output and the actual output. Lots of variations on this theme exist, and many have been successfully used. Back-propagation is a method by which multi-layer networks can implement forms of the delta rule. However, there is a key limitation of this approach: it requires a set of training data which is already classified (in order to present an input pattern with a required output). Many researchers have considered other approaches in order to avoid the need for a pre-classified training set, and several of these approaches have been documented with degrees of success in applications.

This paper explores the use of a particular hybrid approach - the use of a GA to find the link weights of an ANN.

As above, conventionally implemented ANNs are a powerful tool, and are normally used with a training set of data. This is acceptable for many circumstances, but there are other situations where it is not practical and some form of unsupervised training is preferred.

This research considers the use of a GA to generate an ANN to perform a classification function, where:

- a training set is not available
- the proportions of different classes is known *a priori*

... and then as a further stipulation, a human operator would not be available to guide the evolution of the network normally, but maybe could occasionally - after say an extended period of autonomous evolution - select or reject a number of networks.

These assumptions hold in a wide number of potential applications.

It is easiest to illustrate this situation with a simple example. Say the classification is to be between images of apples and bananas. A training set of images pre-classified is not

available. It is known however that 60% of the images are of apples, and 40% are bananas. Having no other information to go on, the GA would evolve networks which did no more than provide a classification function which classified images in a 60:40 split.

It is postulated that this is likely to be an effective way of finding a network to perform a useful classification.

Furthermore, after the GA has found a number of different networks which do indeed classify in a 60:40 split, an operator could then present several images (in any proportion) to each of the networks, and reject those which are not effective in classifying apples and bananas as required. This extra selection can be seen as ensuring that the evolution stays on the required track. It is justified by the observation that it is likely that to manually classify a reasonably large training set of images will be time consuming. However, to look at several images which should all be apples, and notice that a large number of them are bananas, is a quick human operation.

2. Neural Networks

Neural Networks were first developed by McCulloch & Pitts, inspired by biological neural systems. The idea that of computing using a large number of relatively simple units working in parallel as opposed to the conventional paradigm of sequential computing is intuitively attractive - this seems to be how biological brains work.

There are many forms of ANNs. In this experiment the widespread Multilayer Perceptron was used. The key features of these types of network are;

- comprised of individual neuron units, usually implementing a function which comprises a weighted sum of all the inputs
- neuron units organised in layers, with every layer fully interconnected to the next (every neuron in layer x, feeds its result forward to every neuron in layer x+1)

Traditionally the weights associated with the links between each neuron are found by a back-propagation algorithm, which is essentially an error minimisation (hill climbing) algorithm. Whilst this is undoubtedly effective, it assumes that there is a Training Set of pre-classified data samples which can be used to train the network (set the weights) and then a separate testing set. This is known as Supervised learning. (Picton, 1994)

Work has been recently done in the area of semi-supervised or unsupervised learning, and it has been shown (Cohen, 2003) that there situations where adding extra unclassified data samples after supervised training can be beneficial or detrimental - effectively reinforcing accurate or inaccurate training through positive feedback.

3. Genetic Algorithms

GAs provide a powerful way of exploring a complex solution space. They have been applied to both the problems of finding link weights of an ANN, and finding the an effective structure of a network.

Essentially a GA depends on being able to describe a system by a sequence of symbols - by analogy: a chromosome. Different system's chromosomes can be split and combined to create a new generation. Some form of fitness function is then used to select the 'best' individuals systems and these go forward to create the next generation and so on. A random operation is also usually introduced analogous to genetic mutation.

Recent research has been reported with hybrid schemes using GA's to find the link weights of an ANN, and also to find the structure of a network.

A GA will not guarantee to find the optimum solution, or even any solution. However, they tend to be effective in homing in on some effective solution, and are especially useful in extremely large search spaces. They are often called evolutionary algorithms; and although the analogy with biological evolution is strong, it should be realised that the actual mechanisms of biological evolution are far more subtle than these approximations.

4. Experimental Approach

For simplicity, an ANN was implemented on a widespread office spreadsheet programme with the workings of the network realised as formulae within the spreadsheet. The GA was implemented using VisualBASIC routines embedded in the spreadsheet.

The network was structured as follows;

1. Input layer, 14 neural units each giving a weighted sum of 5 pixels
2. First Hidden layer, 5 units with 14 weighted inputs, giving a weighted sum of every one of the input layer units (full interconnection)
3. Second Hidden layer, 5 units with 5 weighted inputs
4. Third Hidden layer, 5 units with 5 weighted inputs
5. Output layer, 1 unit with 5 weighted inputs

A classification function arbitrarily classified the input image as either 'type A' or 'type B' according to whether the output of the output layer was positive or negative.

There was no particular reason for this network architecture being chosen, although it was required to be a fairly complex network in order that the GA would have a large search space (the space encompassing all possible link weights) to work in.

Simple images were presented to the network, each being 12 x 12 grayscale pixels.

Following work by Johnson & Rose, (2005), this following Simon's Three Pixel Principle, it is not considered essential to view - take as input - every single pixel. In this experiment 70 out of the total 144 pixels were chosen as input into the network.

Each ANN was described by a chromosome - a sequence of 140 integers (the x & y co-ordinates of the 70 pixels taken as inputs) and then 195 floating point numbers - the weights of the neuron links.

Twenty such networks were created with random sequences of numbers.

For each network in turn, 50 images were presented and the proportions that that (random) network split the images into were logged.

The 50 images were generated from two 'seed' images according to a probability function p(A), and the copies were processed by adding random noise, modification of the brightness or contrast, or translation 1 pixel in a random direction. Therefore although 48 images were all originally copied from the same 2 seeds, no two of the 50 images were quite the same.

The GA then applied was;

1. Evaluate each network according to the fitness function which simply ranked the 20 networks according to which had achieved a proportional split of the images closest to that known *a priori* as the proportional split between types A & B.
2. Eliminate (delete) the weakest (least fit) 10 network instances
3. Generate a new 10 network instances by picking pairs of the remaining (most fit) networks at random, picking a random split point, and then combining the top half of one network with the bottom half of the other
4. Perform a mutation operation where a percentage of every number in every member of this new generation is subject to a random change

The new generation of network instances was then presented to the same 50 images and the whole process repeated.

4.1 Experimental Results

The following parameters could be used for each network series;

- number of generations to run
- mutation rate
- the proportion split of the images (labelled p(A) as the probability as a % of a sample being type A)
- the variation method of the images

Firstly, 20 experimental runs were performed in order to get some idea on what mutation rate to use, and how many generations it typically took to home in on a working solution

(1) Number of Generations

Performing these runs for 20 generations, with different combinations of the above parameters, gave the results shown in the chart below. The key result shown by the chart y-axis is the mean of the deviations from p(A) of the 10 network instances selected for re-combination.

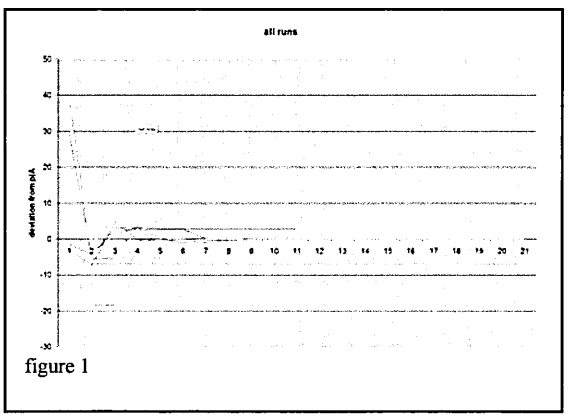


figure 1

Figure 1 indicates empirically that with these parameters and in this context, most network generations home in on an effective solution within 8 generations, with little further improvement seen in the following 12 generations. Therefore it was decided to perform further experiments with 10 generations.

(2) Mutation Factor

Some of the runs were performed with different mutation factors holding other parameters constant. Results for one such collection of runs are shown in figure 2.

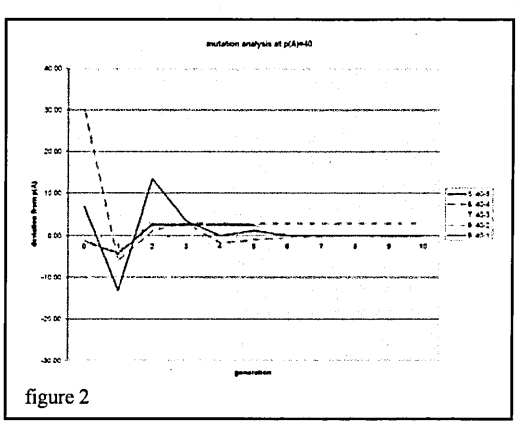


figure 2

Here, 5 runs were performed with the mutation factors 1-5, with P(A) approximately held at 40%. Along with other similar trials, there is no definitive 'best' factor to use, but intuitively it was felt that a 3-5% mutation factor gave reasonable performance.

Further work could be beneficial in determining to a greater extent the optimum number of generations, and mutation factors. For the purposes of these experiments, it was decided to proceed using the figures of 10 generations & 5% mutation.

5. Series 2 Experiments

Here a similar process was carried out, but human interaction took place after 10 generations.

1. 50 images were again populated from 2 seeds, with probability distribution p(A)=36%, and then processed by adding noise

2. 10 generations of the GA were carried out as described above, selecting according to the effectiveness of the network in splitting the images into a proportion close to p(A). This created 10 networks, shown in table 1 below where for each, the % given in the second column represents the split the network achieved in classifying the images. It can be seen that these compare favourably with the 36:64 actual split of the images.

network	% split achieved	errors
1	34%	0
2	40%	1
3	36%	1
4	36%	13
5	32%	7
6	36%	0
7	36%	2
8	34%	10
9	40%	2
10	40%	2

(table 1)

3. a human presented a different set of 50 images to the network - this time processed by adding noise and also modifying the contrast of the images with a blanket factor of 5%, and observed the classification results. The number of classification errors found are shown in the third column of table 1. The networks 4, 5, and 8, were eliminated, and the networks 1 & 6 were copied into their places, thus eliminating networks giving worst performance, and rewarding (by reproducing) networks giving best performance.

(curiously, note that network 4, which gave a perfect 36% split on the initial set of images, performed worst with only slightly different set of images)

4. the process was repeated with this revised set of networks, this time starting with P(A)=18%, and images processed with 10% noise and 5% brightness. The results are shown in table 2 underneath, and it can be seen that after this selection all networks give perfect 18% split, and far fewer errors were found for a different set of test images (this time with 30% noise and -10% contrast).

network	% split achieved	errors
1	18%	1
2	18%	2
3	18%	0
4	18%	1
5	18%	1
6	18%	0
7	18%	0
8	18%	0
9	18%	1
10	18%	0
		(table 2)

Networks 1, 2, and 4 were manually eliminated, and replaced with copies of networks 3, 6, and 7.

5. the process was again repeated with this revised set of networks, this time starting with P(A)=36%, and images processed with 10% noise and a random translation in any direction. The results are shown in table 3 underneath, and it can be seen that after this selection the networks effectiveness is reduced - translations being harder to classify. The number of errors shown with a manual test set are also far greater than previous, that test set also comprising of different a set of translated images coupled with 10% noise factor

network	% split achieved	errors
1	28%	17
2	32%	5
3	28%	17
4	32%	4
5	32%	5
6	32%	5
7	36%	6
8	32%	21
9	32%	22
10	36%	22
		(table 3)

Here networks 1, 3, 8, 9, 10 were eliminated, and replaced with copies of 2, 4, 5, 6

6. the process was again repeated with this revised set of networks, this time starting with P(A)=30%, and images processed with 10% noise and a random translation in any direction. The results are shown in table 4 underneath, and it can be seen that after this the networks effectiveness is improved from the last set, given that this is selecting from a set of translated images. The number of errors shown with a manual test set is also improved than previous, that test set also comprising of different a set of translated images coupled with 10% noise factor

network	% split achieved	errors
1	20%	4
2	20%	4
3	20%	4
4	36%	9
5	20%	4
6	20%	4
7	32%	11
8	36%	9
9	20%	4
10	20%	4
		(table 4)

Here networks 4, 7, 8 were eliminated, and replaced with copies of 1, 2, 3.

7. the process was again repeated with this revised set of networks, with P(A)=20%, and images processed with 10% noise and a random translation in any direction. The results are shown in table 5 underneath, and here the networks effectiveness is again improved. The test set again comprised of different a set of translated images coupled with 10% noise factor.

network	% split achieved	errors
1	20%	4
2	20%	4
3	20%	4
4	20%	4
5	20%	4
6	20%	4
7	20%	4
8	20%	4
9	20%	4
10	20%	4
		(table 5)

The consistency of networks is due largely due to the fact that at this point the cross-generation of the successful networks has led to the proliferation of the successful weights across all networks, meaning that the 10 different networks are virtually the same.

6. Series 3 Experiments

A similar process was carried out to series 2, with all image processing restricted to random direction translations - typically the hardest type of image modification to recognise.

'natural' evolution was carried out for 10 generations, followed by artificial selection - rejection of the worst performing networks. All natural and artificial selection processes were carried out with a different set of images.

After 4 such cycles, the resultant 10 networks offered perfect classification over 5 different sets of images with random translation, and also between 92-94% correct classification for sets of images with random translations coupled with 25% random noise addition - at which point the images were noticeably degraded but still possible for a human to fairly easily identify them.

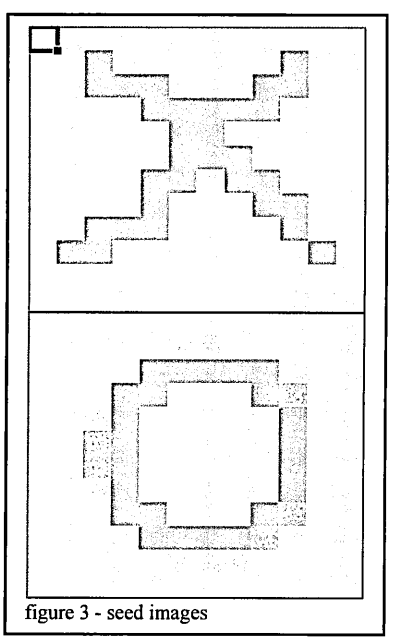


figure 3 - seed images

Figure 3 above shows the two seed images. Figure 4 shows the same images with a random translation and then the addition of 25% noise.

Noise was added according to the following formula: If the noise factor is 25% then every pixel is given a 25% probability that it will be modified by a random value.

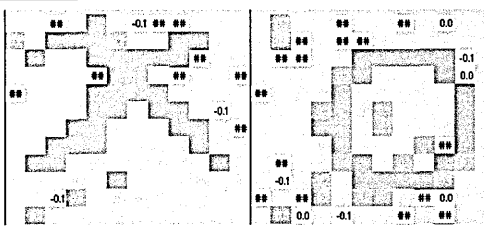


figure 4 - images with 25% noise factor added

Over the course of this experiment the successful networks bred and therefore became copied across the family, and one network came to dominate with 7 out of 10 instances. The retinal pattern - pixels used as inputs - for this successful network is shown in figure 5.

The retinal pattern shown has a cross to represent a pixel which is used as an input to the ANN. Roughly 50% of the image pixels are used as an input, and roughly 50% are ignored.

The retinal patterns are evolved, with the same mechanism as the network weights, and it is intuitively presumed that over evolutionary time the retinal pattern would optimise itself for the context - the particular images presented.

In this case, the retinal pattern presented does not indicate any particular form from which any further conclusions can be drawn, and it is included merely for completeness.

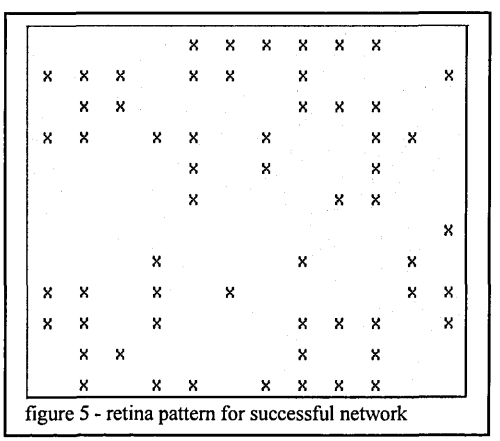


figure 5 - retina pattern for successful network

7. Conclusions

These experiments show empirically and qualitatively that the use of a GA can be effective in generating an ANN to approximate a classification function, which distinguishes between classes, where a training set of pre-classified samples were not available, and the only information given *a priori* is a known proportional split between samples.

The experiments indicate that the GA homes in on a reasonable solution within 10 generations, although this is a highly contextual result and it is not considered that there is any reason to suspect that this would hold more widely.

The longer experimental series furthermore indicate that manual intervention in-between sequences of GA breeding, can enhance performance. This is broadly analogous to the artificial selection used by domestic animal breeders, as opposed to natural selection - normal evolution - set within a context environment. The dog breeder needs no knowledge of how genetics works, but can over time breed dogs with ever longer tails. Here, a human operator needs no knowledge of how the algorithm works, but by throwing out networks which are diverging from the required, and retaining those which appear to be successful in the task, the evolution can be influenced.

This set of experiments investigated a supposed new direction in combining ANNs and GAs, and it is believed to offer a powerful new approach to many classes of problems. This is a very early stage in the development, and several questions for further work are immediately apparent;

i. How does this form of hybrid perform in a wider context with more complex inputs, more variations, and more networks breeding. Would the extra computation involved render this technique nice - but impractical?

ii. Will the artificial selection procedures be actually necessary? or what will be the conditions which would prove that this is beneficial? Is there an optimum level of artificial/natural selection

or will this vary from context to context?

iii. Will the behaviour of a network be something that can be predicted or will it ever only be emergent? With this technique there is a large random element and so any behaviour modelling will have to be based on statistical probability methods - what (if any) accuracy will there be?

iv. In what contexts - if any - will this technique prove useful in preference to other established artificial intelligence techniques?

References

Hornik *et al.* 'Multilayer feedforward networks are universal approximators', *Neural Networks*, 2, 5, 359-366, 1989.

Picton P, *Introduction to Neural Networks*, Macmillan, 1994.

Cohen I, 'Semisupervised learning of classifiers with Application to Human-Computer Interaction', PhD thesis, 2003.

Johnson J, & Rose V, 'Shape Recognition using Randomly Selected Pixel Pair Neurons', Open University, 2005.

Proceedings of the International Symposium on Evolving Intelligent Systems, Plamen Angelov, Dimitar Filev, and Nikola Kasabov (Eds.), at the AISB 2010 convention, 29 March – 1 April 2010, De Montfort University, Leicester, UK

Application of ANN-GA Hybrid to run a conveyor control system

Paul Morley*
Department of Design & Innovation
The Open University
Milton Keynes, MK7 6AA, England
pm4958@tutor.open.ac.uk

Jeff Johnson
Department of Design & Innovation
The Open University
Milton Keynes, MK7 6AA, England

ABSTRACT
This paper describes the use of a genetic algorithm to find the link weights of an Artificial Neural Network used to control an industrial conveyor system, and further proposes a control strategy to allow continual adaptation of the system to changing circumstances thus maintaining the optimization of the system. Results indicate feasibility of this approach, and that performance of the ANN can be as good as conventional systematic controls

Key words
artificial neural networks, evolutionary computation, machine learning, genetic algorithm, self-adapting controls

1. INTRODUCTION
Artificial neural networks (ANNs), and genetic algorithms (GAs), are both well known and widely used in artificial intelligence fields, including their use in controls applications.

ANNs are useful for pattern recognition, and generalizing patterns in complex multi-dimensional space. GAs are useful for searching widely in large solution spaces.

The problem with ANNs, is finding the weights of the links. Various methods have been tried, the most widely known being back-propagation, which is essentially an error minimization (hill climbing) algorithm. However, there is a key limitation of this approach: it requires a set of training data which is already classified (in order to present an input pattern with a required output).
In an industrial context, conventional control systems are often set-up once achieving a reasonable level of performance (not necessarily optimal), and then left forever. However circumstances change, and in the context described here the mix of work coming through the system will drift causing a gradual fall-off in performance. Therefore it was considered that an learning system which could constantly adapt to the circumstances would be beneficial.

This paper explores the use of a hybrid system for controlling an industrial conveyor system. A single ANN is used as a black-box decision maker about the next operation to be executed, and a GA is used to find the link weights of a population of ANNs.

2. NEURAL NETWORKS
Neural Networks were first developed by McCulloch & Pitts, inspired by biological neural systems. The idea that of computing using a large number of relatively simple units working in parallel as opposed to the conventional paradigm of sequential computing is intuitively attractive.

There is also proof that in theory at least, a 2 layer network is able to implement any measurable function to an arbitrary degree of accuracy. This important result was first established by Kolmogorov (the Kolmogorov existence theorem is actually not very useful in practice, because of the unknown nature of the functions used by each unit) and then in a manner more directly relevant to ANN development by Hornik et al. [1]. Although these results show that a 2 layer (i.e. 1 hidden layer) network could in theory implement any function, in practice it may be the case that a more efficient implementation could be found by using more layers.

There are many forms of ANNs. In this experiment the a fairly simple feed-forward network was used with 3 processing layers. This network comprises layers of individual units fully interconnected to the next layer, each unit implementing a non-linear sigmoid function of a weighted sum of all the inputs

Traditionally the weights associated with the links between each neuron are found by a back-propagation algorithm, which is essentially an error minimization (hill climbing) algorithm. Whilst this is undoubtedly effective, it assumes that there is a Training Set of pre-classified data samples which can be used to train the network (set the weights) and then a separate testing set. This is known as Supervised learning. [2]

3. GENETIC ALGORITHMS

GAs provide a powerful way of exploring a complex solution space. Essentially a GA depends on being able to describe a system by a sequence of symbols - by analogy: a chromosome. Different system's chromosomes can be split and combined to create a new generation. Some form of fitness function is then used to select the 'best' individuals systems and these go forward to create the next generation and so on. A random operation is also usually introduced analogous to genetic mutation.

In essence, Genetic Algorithms tend to include the following features;

- a population of multiple instances of trial solutions to the problem in hand
- some way of determining the relative fitness or effectiveness of each solution
- some operator(s) which produce new solution instances based on the more effective previous instances.

There are lots of variations to this approach; the operators used can vary - for example, mutation & recombination are two common operators inspired by biological genetics, but are not the only ones possible. Equally, there are many different approaches for managing the population - the strongest instances in one generation may be retained in the next, or the entire population may be replaced with new instances - to name just two possibilities.

A GA will not guarantee to find the optimum solution, or even any solution. However, they tend to be effective in homing in on some effective solution, and are especially useful in extremely large search spaces. They are often called evolutionary algorithms; and although the analogy with biological evolution is strong, the actual mechanisms of biological evolution are far more subtle than these approximations.

4. HYBRID METHODS

Genetic Algorithms have been applied to both the problems of finding link weights of an ANN, and finding the an effective structure of a network with researchers reporting schemes which alternatively evolve only the structure of the network, leaving the problem of finding the weights to a deterministic approach - such as classical back-propagation: a gradient descent optimization algorithm, or keeping the structure fixed and evolving the weights. [3] provides a general review of this field.

For example, [4] describes an algorithm used to evolve both the weights and the structure of ANNs. See [5] for an earlier example which also used evolutionary programming to evolve both the weights and structure. This latter refers specifically to a system where the selection process acted on only the output of the algorithm and not on the ideas underlying the output. That this is effective and efficient is contrary to the view of [6]. [7] and [8] give further examples of the use of an GA with selection based only on overall performance.

5. INDUSTRIAL CONTEXT

The system was applied to control a conveyor system in the context of an industrial laundry to a set of dryers. The general layout is shown in figure 1.

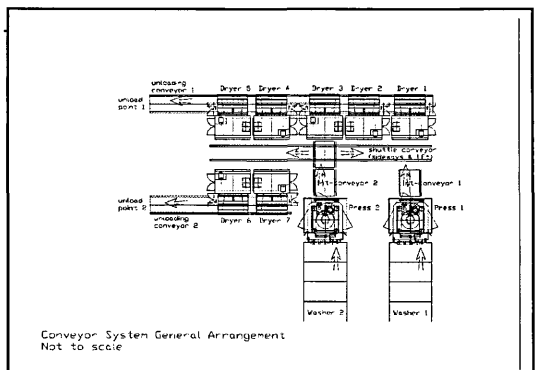


fig 1: General layout of conveyor system

System Description

This is a commercial laundry installation processing up to 6000kg of hospital linen per hour. The system was commissioned in November 2005, in a private facility in Ireland.

Batches of work move through the system, from the washers, to the water-extraction presses, one-batch storage conveyor, and into one of the dryers. The performance is affected by the shuttle conveyor which takes batches of work from one of the wash-lines, and takes them to one of the dryers. Performance is affected due to the transit time of the shuttle - for example:

- if the shuttle is in position for dryers 5 & 6, and a load of linen becomes available from washer 1, then washer 1 cannot unload until the shuttle has moved to that position thus losing that time for the washer.
- some types of work take longer in the dryers than others. If a shuttle puts long-drying work into a dryer close to the washers, then it ties up that dryer for a long time, hence dryers further away have to be used leading to longer transit times.

Generally best performance is achieved by putting longer dry-time loads in dryers further from the washers, and then parking the shuttle in front of the next washer to unload ready for it. This is the basis of conventional control algorithm. More subtle effects still cause sub-optimal performance, such as the tie-up

of the dryer unload conveyors caused by loads from - for example - dryers 1, 2 & 3 holding up the unloading of dryers 4 & 5.

Optimal utilization would achieve 30 batches from each washer per hour. Conventional control typically achieves 85-90% of this. The mix of work (with different proportions of batches with different dry times) varies over time and this causes a variation of performance.

6. HYBRID CONTROL ARCHITECTURE

A new control paradigm is proposed which has two phases:

- initial set-up
  A population of ANNs is generated and a GA is used to evolve to the point where the most successful ANN can be used to run the system. This is analogous to initial commissioning of the system and can be done quite quickly simulating the system on a computer.

  This paper describes work on this phase only.

- adaptive running
  The controlling ANN runs the system but a population of alternative ANNs is maintained evolving with live data. When an ANN from this population consistently achieves better results than the current controlling ANN, it supplants it. In this way the control will constantly adapt to changing circumstances

Some advantages of this architecture over conventional controls are;

- conventional controls require a full analysis of the operation of the controlled system. In this case this is possible but may not always be so
- conventional controls do not adapt to changing circumstances (the initial analysis may not hold for changed circumstances)
- the method is relatively easy to implement, and also takes very little adaptation to apply to control systems in other contexts.

However the disadvantages include;

- optimal control may not be attained, and without the system analysis, the optimal performance may not be known
- the operations of the system cannot be easily explained, could not be guaranteed so would not be suitable for safety critical systems

7. APPLICATION

A population of 50 ANNs was generated, each with 2 hidden layers of 28 & 13 units. The inputs to the ANN were parameters representing the state of the system - for example - a number representing the current load in the press, on the conveyor etc.

Each ANN was represented as a sequence of its link weights, and the GA was applied to this sequence on the following algorithm;

- run each ANN in turn for a simulated 2 hours operation and measure performance (loads produced)
- always keep best performing ANN
- all ANNs with less than average performance, eliminate from the population
- replace eliminated ANNs with new one generated by single point crossover from two parent ANNs each with better than average performance. Also apply a small mutation factor.

Fitness Function

To determine the performance of the ANN, a simple measure of how many batches the system had produced in the timescale was used. This method of measuring is remote from the individual decision outputs from the ANN. The key point is that the optimum output of the ANN for each individual decision made is not known without encountering the disadvantages discussed earlier. However the overall production of the system is known and furthermore is the parameter the operator of the system really needs to maximize.

Note that if the optimum output of the ANN was known for each decision, then simple gradient based learning would be appropriate. However, the optimum output could only generally be obtained following a full analysis of the system and this carries the disadvantages mentioned earlier.

8. ISLANDS

It was decided to implement an island strategy for the GA. This follows the parallel population strategy of [10]. This was carried over several phases;

phase 1 5 populations of 50 ANNs were evolved in isolation for 100 generations, with differing mutation rates in each population

phase 2 2 populations of 50 ANNs were evolved in isolation for 100 generations. Each population was formed from individuals mixed from the phase 1 populations

phase 3 1 population of 50 ANNs taken from a mix from the phase 2 populations. Evolution was run for 400 generations.

phase 4 the phase 3 generation was run for a further 500 generations, with a mutation rate steadily declining from 0.5% down to 0.1%

9. RESULTS & DISCUSSION

Figure 2 shows the increase in performance over the series, with a measure taken at the end of each set of 100 generations. The two curves show the performance of the best individual network in the final generation of the run, and the average of all networks over the final generation.

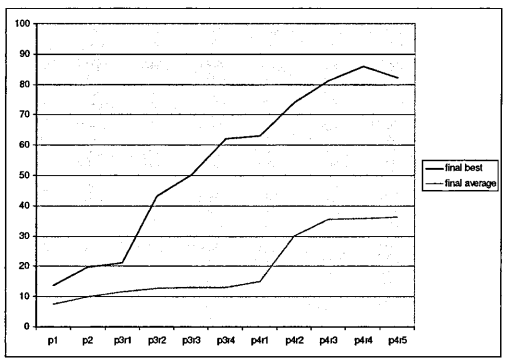


fig 2: performance (number of batches produced in the test time) of final best and final average ANNs for each evolutionary phase

The vertical axis shows the number of batches produced, over the trial period which was 5400 seconds (1.5 hours). The theoretical maximum production in this time is 90 batches It can be seen that after the various phases of evolution, the best performing networks are able to reach performances of well over 80, which is a very respectable system utilization. (Conventional control typically achieves 85-90% of this, or 76-81 batches).

Care must be taken not to overstate these results - although the evolved controls achieve a best here of 85 batches, on occasions (due to the mix of work coming through the system) the conventional control can achieve the maximum 90 batches. This is a stochastic system. However this does indicate the credibility of evolution for generating ANN system controls, which can be comparable to conventional methods.

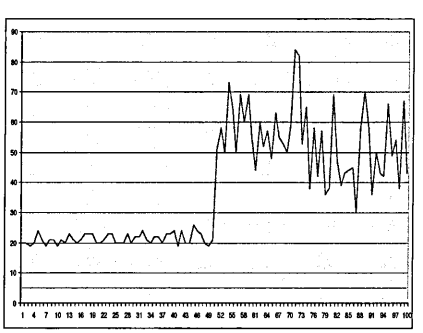


fig 3: performance (number of batches produced in the test time) of best, average, and worst ANNs.

Figure 3 shows the performance of the system during the 100 generations of phase 3 run 2. The top curve shows the best performing network in each generation. The lower curves show the average performance and the worst performance respectively. There is a step change from generation 50, (best performing network produced 21 batches), and generation 51, (best performing network produced 51 batches). Prior to this the performance had been relatively static. After this point, the best-performing network performance was relatively stable while the average performance climbed - as the successful elements of the best performing network propagated through the population. It is clear that the GA had jumped to a new and better part of the search space.

This particular result is a striking illustration of the random nature of evolutionary strategies and their ability to jump into a new area of the search space avoiding local minima.

10. REFERENCES

[1] Hornik K, Stinchcombe M, White H, 'Multilayer Feedforward Networks are Universal Approximators', Neural Networks 2 pp359-366, 1989

[2] Picton P, Introduction to Neural Networks, Macmillan, 1994.

[3] Yao X, 'Evolving Artificial Neural Networks', Proc. IEEE. Vol. 87 no. 9. 1999

[4] Curran D, O'Riordan C, 'Evolving Connect-Four Playing Neural Networks Using Cultural learning'. 2004.
http://ww2.it.nuigalway.ie/cirg/publications.html

[5] Fogel D, 'Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe'. Proc. Amer. Power Conf. 1993 pp875-879

[6] Penrose R, 'The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics'. Penguin Books. 1989

[7] Morley P, ' training a genetic algorithm and neural network hybrid pattern classifier by population statistics', AISB conf. 2005

[8] Morley P, 'evolving neural networks to play noughts & crosses', ISKE conf. 2009.

[9] Wolpert D H and Macready W G, "No free lunch theorems for optimization," IEEE Trans. Evolutionary Computation, vol. 1, pp. 67–82, Apr. 1997

[10] Whitley D, Rana S, and Heckendorn R. 'Island Model Genetic Algorithms & Linearly Separable Problems'. Journal of Computing and Information Technology. 1998.

Evolving Neural Networks To Play Noughts & Crosses

Paul Morley
Department of Design & Innovation
The Open University
Milton Keynes, MK7 6AA, England
pm4958@tutor.open.ac.uk

1. Abstract

A set of experiements were performed to investigate the effectiveness of using a Genetic Algorithm (GA) to evolve a competitive population of Artificial neural networks (ANNs) to play the game Noughts & Crosses. The input to the network were pairs of game positions that were precursors to a win. The effect of different parameters was investigated, by changing those parameters - namely - payoff function, harshness of the environment, and chance of playing a random player instead of another ANN. Results indicated a particular set of parameters were more effective than others although this is not asserted to be a general result.

2. Introduction

Artificial neural networks (ANNs), and genetic algorithms (GAs), are both well known and widely used in artificial intelligence fields, including their use in controls applications.

ANNs are useful for pattern recognition, and generalizing patterns in complex multi-dimensional space. GAs are useful for searching widely in large solution spaces.

The problem with ANNs, is finding the weights of the links. Various methods have been tried, the most widely known being back-propagation, which is essentially an error minimization (hill climbing) algorithm. However, there is a key limitation of this approach: it requires a set of training data which is already classified (in order to present an input pattern with a required output).

This paper explores the use of a hybrid system in the context of playing the game noughts & crosses (tic-tac-toe). An ANN is used as a 'controlling mind' to make decisions about the next move to make, and a GA is used to find the link weights of a population of ANNs.

3. Neural Networks

Neural Networks were first developed by McCulloch & Pitts, inspired by biological neural systems. The idea that of computing using a large number of relatively simple units working in parallel as opposed to the conventional paradigm of sequential computing is intuitively attractive.

There is also proof that in theory at least, a 2 layer network is able to implement any measurable function to an arbitrary degree of accuracy. This important result was first established by Kolmogorov (the Kolmogorov existence theorem is actually not very useful in practice, because of the unknown nature of the functions used by each unit) and then in a manner more directly relevant to ANN development by Hornik et al. [1]. Although these results show that a 2 layer (i.e. 1 hidden layer) network could in theory implement

any function, in practice it may be the case that a more efficient implementation could be found by using more layers.

There are many forms of ANNs. In this experiment the a fairly simple feed-forward network was used with 3 processing layers. This network comprises layers of individual units fully interconnected to the next layer, each unit implementing a non-linear sigmoid function of a weighted sum of all the inputs

Traditionally the weights associated with the links between each neuron are found by a back-propagation algorithm, which is essentially an error minimization (hill climbing) algorithm. Whilst this is undoubtedly effective, it assumes that there is a Training Set of pre-classified data samples which can be used to train the network (set the weights) and then a separate testing set. This is known as Supervised learning. [2]

4. Genetic Algorithms

GAs provide a powerful way of exploring a complex solution space. Essentially a GA depends on being able to describe a system by a sequence of symbols - by analogy: a chromosome. Different system's chromosomes can be split and combined to create a new generation. Some form of fitness function is then used to select the 'best' individuals systems and these go forward to create the next generation and so on. A random operation is also usually introduced analogous to genetic mutation.

In essence, Genetic Algorithms tend to include the following features;

- a population of multiple instances of trial solutions to the problem in hand
- some way of determining the relative fitness or effectiveness of each solution
- some operator(s) which produce new solution instances based on the more effective previous instances.

There are lots of variations to this approach; the operators used can vary - for example, mutation & recombination are two common operators inspired by biological genetics, but are not the only ones possible. Equally, there are many different approaches for managing the population - the strongest instances in one generation may be retained in the next, or the entire population may be replaced with new instances - to name just two possibilities.

A GA will not guarantee to find the optimum solution, or even any solution. However, they tend to be effective in homing in on some effective solution, and are especially useful in extremely large search spaces. They are often called evolutionary algorithms; and although the analogy with biological evolution is strong, the actual mechanisms of biological evolution are far more subtle than these approximations.

5. Hybrid Methods

Genetic Algorithms have been applied to both the problems of finding link weights of an ANN, and finding the an effective structure of a network with researchers reporting schemes which alternatively evolve only the structure of the network, leaving the problem of finding the weights to a deterministic approach - such as classical back-propagation: a gradient descent optimization algorithm, or keeping the structure fixed and evolving the weights. See [3] for a general review of this field.

For example, [4] describes an algorithm used to evolve both the weights and the structure of ANNs. See [7] for an earlier example which also used evolutionary programming to evolve both the weights and structure. This latter is particularly interesting as it refers specifically to a system where the selection process acted on only the output of the algorithm and not on the ideas underlying the output. A point made by Fogel in this paper is that this is effective and efficient, contrary to the view put forward by Penrose in [5].That this is possible - to use evolutionary strategies for practical purpose without any knowledge of the process beyond

the desired result - is a view central to this current programme of research. Another example where use of an GA with selection only based on a high level metric was reported in [6].

This paper reports a set of experiments following on from work reported in [4]. There the inputs to the network were the states of the individual spaces. Here the inputs were pairs of the spaces.

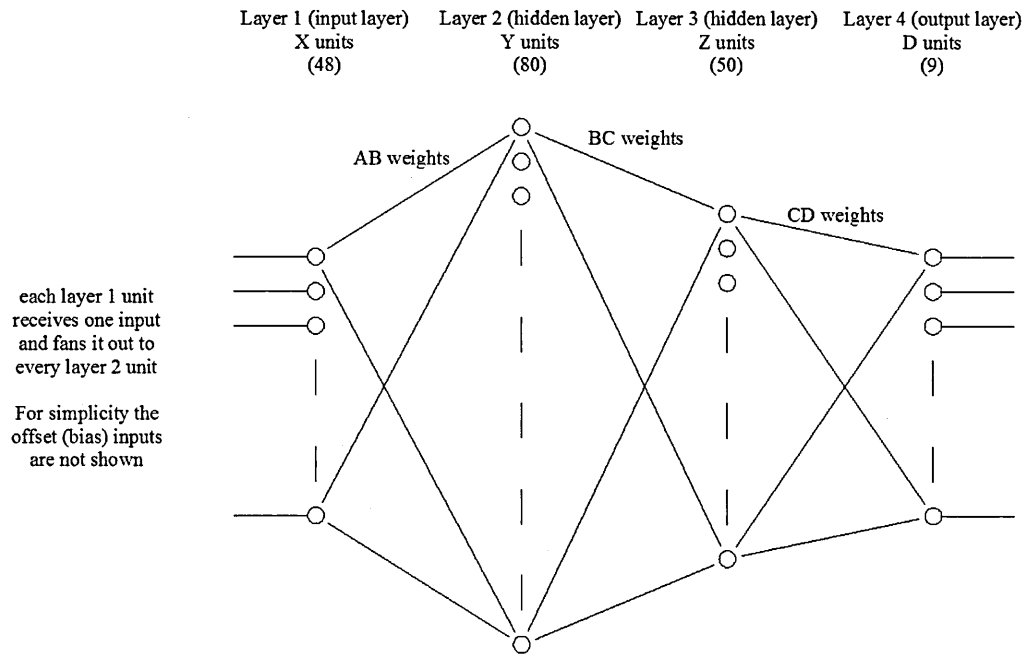
6. Method

Noughts & Crosses (Tic-tac-toe) is a traditional game of simple strategy. Two players attempt to get a line of three counters taking alternate turns on a 3x3 board.

The game was implemented on a VB simulator, and a population of ANNs generated as follows;

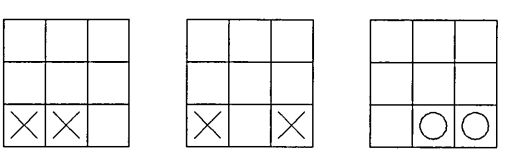
1. Static Parameters	
ANN	structure - fully interconnected feed-forward 4 layer, as shown figure 1 48 inputs 80 hidden units layer 1 50 hidden units layer 2 9 outputs (8290 link-weights in total) transfer - weighted sum including bias, with sigmoid function
GA	population size 50 generations 2000 mutation rate 1% x (0.9)^(no generations) recombination by single point crossover, random split point parents - selected randomly from pool of >average performance
performance assessment	each network taken in turn to play 10 games, each against a randomly selected opponent from the ANN population, or the possibility to play a deterministic automatic player instead of another ANN Payoff function = {variable} (score awarded for each loss, draw, or win.) For each generation each ANN is given a score comprising the sum of payoffs for each of the 10 games. This score is used in the GA strategy.
GA strategy	calculate average performance and find best performer - best performer is always retained for each ANN, if performance < average, then variable chance of replacement replacement is by single point crossover, with random crossover point, using two randomly selected ANN parents from the pool of ANNs having better than average performance
2. Variables	
ANN	link weights - initialized with random weights

GA	payoff function {1,-10,0}, {10,-1,0} chance of replacement if score < average, 50%, 95% chance of playing deterministic player, 50%, 95%



(figure 1)

The 48 inputs comprised pairs of game-spaces, for example;



(figure 2)

Figure 2 shows three of the 48 permutations of states which precede a win (or lose). Each of the 48 win-precursor states was given as an input to the network. The first two illustrated were assigned a value of 1, and the third a value of -1. It was necessary that the third space was empty, but the remaining spaces were 'don't cares'. Previous experiments (not reported here) had carried out similar work without the use of win-precursor inputs. Results were in fact similar to those reported here.

Deterministic player

There was also the chance of playing the deterministic player which operated as follows;

- if a win can be obtained then go there
- otherwise go in a random location

7. Results

8 sets of experiments were conducted. At the end of each run, the networks in the population were tested against a test player that operated ;

- ANN moves first, first replying move taken in each of the remaining places in turn, then;
  - if a win can be obtained then go there
  - if a win for the opponent can be blocked the go there
  - otherwise go in a random location

Thus 9 results were obtained for each network. The table below records the number of those games which were won or lost by the network, or drawn. Furthermore for comparison a randomized set of ANNs were also tested

run	payoff function	probability: replacement if <average	probability: playing auto-player	win	lose	draw	%win	%lose
1	{10,-1,0}	0.95	0.95	40	298	112	8.89%	66.22%
2	{1,-10,0}	0.95	0.95	16	299	135	3.56%	66.44%
3	{10,-1,0}	0.50	0.95	36	328	86	8.00%	72.89%
4	{1,-10,0}	0.50	0.95	29	286	135	6.44%	63.56%
5	{10,-1,0}	0.95	0.50	30	323	97	6.67%	71.78%
6	{1,-10,0}	0.95	0.50	32	292	126	7.11%	64.89%
7	{10,-1,0}	0.50	0.50	58	270	122	12.89%	60.00%
8	{1,-10,0}	0.50	0.50	34	268	148	7.56%	59.56%
random	n/a	n/a	n/a	24	311	115	5.33%	69.11%

The following table re-orders the results with most successful (highest win %) first:

run	payoff function	probability: replacement if <average	probability: playing auto-player	win	lose	draw	%win	%lose
7	{10,-1,0}	0.50	0.50	58	270	122	12.89%	60.00%
1	{10,-1,0}	0.95	0.95	40	298	112	8.89%	66.22%
3	{10,-1,0}	0.50	0.95	36	328	86	8.00%	72.89%
8	{1,-10,0}	0.50	0.50	34	268	148	7.56%	59.56%
6	{1,-10,0}	0.95	0.50	32	292	126	7.11%	64.89%
5	{10,-1,0}	0.95	0.50	30	323	97	6.67%	71.78%
4	{1,-10,0}	0.50	0.95	29	286	135	6.44%	63.56%
random	n/a	n/a	n/a	24	311	115	5.33%	69.11%
2	{1,-10,0}	0.95	0.95	16	299	135	3.56%	66.44%

8. Conclusions

It was supposed that evolution as a general approach will tend to solve problems such as this, where no information regarding the object of the game or training as to appropriate moves was given. Given the

present state of the board, the ANN had to decide on the best next state, and no feedback was given until the end of the game, at which point feedback was limited to 'win / draw / lose' as outlined.

It was further supposed that while the evolutionary process is robust, there are parameters which will affect the speed of convergence to an optimum. Three parameters were varied in the course of this experiment. There are - as the table in section 5 indicates - many more parameters, and it would be interesting to establish their relative effect.

Payoff function

The payoff function for win / draw / lose is important, and in particular that it is asymmetric. The actual figures are irrelevant, but the relative values are.

Two payoff functions were tested with the following expectation;

{10, -1, 0} :	strong advantage for winning	- expect higher proportion to win
{1, -10, 0} :	strong disadvantage for losing	- expect higher proportion to win or draw

From the ordered table it can be seen that the first payoff resulted in the more successful individuals.

Probability of replacement if below average performance

This represents the harshness of the environment. The probability that an individual will be eliminated from the population if its performance is below average.

Two probabilities were tested with the following expectation;

0.95 : strong chance of elimination
0.50 : weaker chance of elimination

From the ordered table it can be seen that the weaker chance of elimination resulted in the more successful individuals - perhaps because a harsh environment eliminates individuals too soon - before they have chance to develop successfully.

Probability of playing the deterministic player

The deterministic player will always take direct opportunity to win. Therefore we would expect that the ANNs with more exposure to this player, to be more effective at blocking wins.

Two probabilities were tested with the following expectation;

0.95 : strong chance of playing deterministic
0.50 : weaker chance of playing deterministic

The results on this did not give a consistent indication that this parameter was important here.

9. End note

As reported in [3] the best approach to problem optimization is problem dependent - according to the no free lunch theorem [8], and this also holds for the initial conditions and parameters. The best parameters for one problem will not necessarily be the best for another.

Overall the results reported here were disappointing. Although evolution did produce activity which was markedly better than random, it was not as effective as was hoped. Furthermore, the use of pairs (win-precursor states) was expected to be more effective than simple inputs but this did not appear to be the case.

10. References

[1] Hornik K, Stinchcombe M, White H, 'Multilayer Feedforward Networks are Universal Approximators', Neural Networks 2 pp359-366, 1989

[2] Picton P, Introduction to Neural Networks, Macmillan, 1994.

[3] Yao X, 'Evolving Artificial Neural Networks', Proc. IEEE. Vol. 87 no. 9. 1999

[4] Fogel D, 'Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe'. Proc. Amer. Power Conf. 1993 pp875-879

[5] Penrose R, 'The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics'. Penguin Books. 1989

[6] Morley P, ' training a genetic algorithm and neural network hybrid pattern classifier by population statistics', AISB conf. 2005

[7] Curran D, O'Riordan C, 'Evolving Connect-Four Playing Neural Networks Using Cultural learning'. 2004. http://ww2.it.nuigalway.ie/cirg/publications.html

[8] Wolpert D H and Macready W G, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Computation*, vol. 1, pp. 67–82, Apr. 1997

CELTIC LINEN, DRINAGH
hotel shift
piece mix agreed during meeting (27-10-2004)

total wash capacity 3,531 kg/hour
surplus 76 kg/hour

key plant data

330,000 net pieces per week
39 hours run per week

work breakdown

	% by pieces	average weight	dry code	net pieces per hour	net weight per hour
total	**100.0%**	**0.408**		**8,462**	**3,454.4**
sheets	15.00%	1.000	none	1,269	1,269.2
pillow cases	20.00%	0.200	part	1,692	338.5
towels	32.00%	0.350	full	2,708	947.7
napkins	20.00%	0.100	none	1,692	169.2
table linen small	2.25%	0.500	none	190	95.2
table linen large	2.25%	1.200	none	190	228.5
coloured table linen	1.00%	0.750	none	85	63.5
tea towels	3.00%	0.100	part	254	25.4
duvet covers (large)	1.00%	1.500	part	85	126.9
duvet covers (single)	1.00%	1.000	part	85	84.6
misc	2.50%	0.500	full	212	105.8
				0	0.0

washing machine 1 ☑ present?
batch kg 75 length 12 cycle 130 sec
utilisation factor 85% this machine will produce 23.53846 batches/hour
1,765 kg/hour
with a wash time of 26.0 minutes

washing machine 2 ☑ present?
batch kg 75 length 12 cycle 130 sec
utilisation factor 85% this machine will produce 23.53846 batches/hour
1,765 kg/hour
with a wash time of 26.0 minutes

washing machine 3 ☐ present?
50 16 100
85%

assumptions: no rewash, work mixed in each washer

dryers

batches/hour to dry 55.38462
(worst case, 100%)

for each dry code	none	part	full
dry time (min)	1	6	20
% of work	52.8%	16.7%	27.4%
batches/hour	29	9	15
dryer-min	29.3	55.4	303.9
dryers	0.5	0.9	5.1

total dryers required 7

staffing calculation

pieces per operator hour 188.0
total staff 45
overhead 0
net total staff 45

soiled facilitating 2
soiled reception 2
sorting 6
conveyor packing 2
despatch 3

finishing equipment

allocation of % of each work type to finisher

	reference pieces per hour	utilisation 80% actual	staff per machine	sheets 1,269	pillow cases 1,692	towels 2,708	napkins 1,692	table linen smaable linen larg	coloured table lir 85	tea towels 254	net covers (lar et covers (sin 85	misc 212	0 0	total pieces/hr for these machines	machines required	machines actual	staff total
ironers																	
EMS sheet line	850	680	2											0	0.0	1	2
Jensen Duplex sheets	850	680	4											0	0.0	1	4
GEM mixed line	1000	800	4											0	0.0	1	4
small piece line	525	420	1.5											0	0.0	4	7
Stahl misc line	300	240	5											0	0.0	1	6
folders																	
TFS from Carlow	750	600	1											0	0.0	1	1
Jensen towel folder	650	520	1											0	0.0	1	1
reloc Jensen Tematic	700	560	1											0	0.0	1	1
Jensen blanket	350	280	2											0	0.0	2	4

Cannot be calculated, as the work mix has now changed:
- table linen will now be put through Drinah unit 1